

INTEGRATING STRUCTURED DATA ON THE WEB

by

Thanh Hoang Nguyen

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computing

School of Computing

The University of Utah

May 2013

Copyright © Thanh Hoang Nguyen 2013

All Rights Reserved

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of Thanh Hoang Nguyen

has been approved by the following supervisory committee members:

<u>Juliana Freire</u>	, Chair	<u>10/15/2012</u> Date Approved
<u>Suresh Venkatasubramanian</u>	, Member	<u>9/28/2012</u> Date Approved
<u>Cláudio T. Silva</u>	, Member	<u>10/15/2012</u> Date Approved
<u>Renee J. Miller</u>	, Member	<u>11/09/2012</u> Date Approved
<u>Viviane Moreira</u>	, Member	<u>10/29/2012</u> Date Approved

and by Alan Davis, Chair of
the Department of School of Computing

and by Charles A. Wight, Dean of The Graduate School.

ABSTRACT

The explosion of structured Web data (e.g., online databases, Wikipedia infoboxes) creates many opportunities for integrating and querying these data that go far beyond the simple search capabilities provided by search engines. Although much work has been devoted to data integration in the database community, the Web brings new challenges: the Web-scale (e.g., the large and growing volume of data) and the heterogeneity in Web data. Because there are so much data, scalable techniques that require little or no manual intervention and that are robust to noisy data are needed. In this dissertation, we propose a new and effective approach for matching Web-form interfaces and for matching multilingual Wikipedia infoboxes. As a further step toward these problems, we propose a general prudent schema-matching framework that matches a large number of schemas effectively. Our comprehensive experiments for Web-form interfaces and Wikipedia infoboxes show that it can enable on-the-fly, automatic integration of large collections of structured Web data. Another problem we address in this dissertation is schema discovery. While existing integration approaches assume that the relevant data sources and their schemas have been identified in advance, schemas are not always available for structured Web data. Approaches exist that exploit information in Wikipedia to discover the entity types and their associate schemas. However, due to inconsistencies, sparseness, and noise from the community contribution, these approaches are error prone and require substantial human intervention. Given the schema heterogeneity in Wikipedia infoboxes, we developed a new approach that uses the structured information available in infoboxes to cluster similar infoboxes and infer the schemata for entity types. Our approach is unsupervised and resilient to the unpredictable skew in the entity class distribution. Our experiments, using over one hundred thousand infoboxes extracted from Wikipedia, indicate that our approach is effective and produces accurate schemata for Wikipedia entities.

CONTENTS

ABSTRACT	iii
LIST OF FIGURES	vii
LIST OF TABLES	ix
ACKNOWLEDGEMENTS	x
CHAPTERS	
1. INTRODUCTION	1
1.1 Motivation	1
1.2 Technical Contributions and Outline	4
2. MATCHING WEB-FORM INTERFACES	7
2.1 Introduction	7
2.2 Problem Definition and Solution Overview	10
2.2.1 Problem Definition	10
2.2.2 Solution Overview	11
2.3 Form Matching	12
2.3.1 Computing Similarities	12
2.3.1.1 Label Similarity	12
2.3.1.2 Domain-Value Similarity	14
2.3.1.3 Correlation	14
2.3.2 Matching Discovery	15
2.3.2.1 Combining the Similarities	15
2.3.2.2 Prudent Matcher	16
2.3.2.3 Integrate Matches	17
2.3.3 Matching Growth	20
2.4 Experimental Evaluation	21
2.4.1 dataset and Evaluation Metrics	21
2.4.1.1 dataset	21
2.4.1.2 Evaluation Metrics	22
2.4.2 Effectiveness of FormMatch	23
2.4.2.1 Comparing Different Combinations	24
2.4.2.2 Syntactic Merging	24
2.4.3 Threshold Sensitivity	24
2.4.4 Large Dataset with Automatic Label Extraction	25
2.4.4.1 Adjusted Evaluation	27
2.5 Related Work	28
2.6 Discussion	32

2.6.1	Noise and Rare Attributes	32
2.6.2	Limitation	33
2.6.3	Future Work	34
3.	MATCHING MULTILINGUAL SCHEMATA IN WIKIPEDIA	35
3.1	Introduction	35
3.2	Problem Definition and Solution Overview	38
3.2.1	Problem Definition	38
3.2.2	Solution Overview	39
3.3	WikiMatch	40
3.3.1	Computing Cross-Language Similarities	40
3.3.1.1	Cross-Language Value Similarity	40
3.3.1.2	Link Structure Similarity	40
3.3.1.3	Attribute Correlation	41
3.3.2	Deriving Correspondences	43
3.3.3	Revising Uncertain Matches	45
3.4	Experimental Evaluation	46
3.4.1	Dataset and Evaluation Metrics	46
3.4.1.1	Dataset	46
3.4.1.2	Evaluation Metrics	47
3.4.2	Comparison against Existing Approaches	48
3.4.2.1	Weighted Precision and Recall	48
3.4.2.2	Precision and Recall Curves	51
3.4.2.3	Comparing with Consistency Learning	51
3.4.2.4	Macro-averaging	53
3.4.3	Contribution of Different Components	53
3.4.4	Exploring Different Alternatives for Attribute Correlation	53
3.4.5	Threshold Sensitivity	54
3.5	Case Study: Evaluating Cross-language Queries	55
3.6	Related Work	57
3.6.1	Cross-Language Ontology Alignment	57
3.6.2	schema-matching	58
3.6.3	Cross-Language Infobox Alignment	58
3.6.4	Cross-Language Concept Alignment	59
3.7	Conclusion	59
4.	A GENERAL PRUDENT SCHEMA-MATCHING FRAMEWORK	61
4.1	The PruSM Matching Framework	61
4.1.1	Prudent Matcher	62
4.1.2	Matching Algorithm	64
4.2	Customize PruSM for Different Case Studies	65
4.2.1	Customizing PruSM	65
4.2.1.1	Customize the Correlation Measure	65
4.2.1.2	Customize the Supporting Similarities	66
4.2.1.3	Customize the Prudent Matcher	66
4.2.1.4	Customize the Matching Components	66
4.2.2	Summary of Results	66
4.2.2.1	Outperform Existing Approaches	66
4.2.2.2	Different Combining Strategies	66
4.2.2.3	Contribution of Different Features	67
4.2.2.4	Threshold Sensitivity	68
4.3	Conclusion and Discussion	68

5. ORGANIZING WIKIPEDIA BY CLUSTERING INFOBOXES	70
5.1 Introduction	70
5.2 Problem Definition and Solution Overview	75
5.2.1 Problem Definition	75
5.2.2 Terminology and Definitions	75
5.2.3 Solution Overview	75
5.3 Clustering Wikipedia Infoboxes (WIClust)	77
5.3.1 Attribute Clustering	77
5.3.2 Infobox Grouping	80
5.3.3 Efficiency	81
5.3.4 Cluster Reconciliation	82
5.4 Semantic Refinement and Synonyms Identification	83
5.5 Experimental Evaluation	85
5.5.1 Dataset and Evaluation Metrics	85
5.5.1.1 Dataset	85
5.5.1.2 Evaluation Metrics	86
5.5.2 Quality of the Entity Clusters	87
5.5.2.1 Comparing against Different Clustering Methods	87
5.5.2.2 Comparing with DBpedia	90
5.5.2.3 Covering New Templates and Discovering New Entity Types	91
5.5.3 Reconciliation and Semantic Refinement	91
5.5.3.1 Reconciled Entity Clusters	91
5.5.3.2 Synonyms and Semantic Refinement	92
5.5.4 Attribute Clusters	92
5.6 WikiQuery: A Case Study	94
5.7 Related Work	96
5.8 Conclusion	100
6. CONCLUSION AND FUTURE WORK	101
6.1 Conclusion	101
6.2 Future Work	103
REFERENCES	106

LIST OF FIGURES

1.1	Information aggregators integrate information from multiple sources and provide a unified interface that allows user to query this information. Using Yahoo! Autos, for example, users can search for cars that are advertised all over the United States.....	1
1.2	Structured information on the Web is available not only in the Deep Web, hidden behind Web-form interfaces, but it is also published on the Web surface as tables (a) and records with implicit structure(b).....	2
1.3	To support faceted queries or to compare products advertised by different vendors, product search engines must identify correspondences between the different schemata they use. For example, for laptops, different terms are used to represent the CPU used (Chipset, Processor)	3
1.4	The availability of structured information in the form of infoboxes makes it possible to answer complex queries such as <i>Find the titles and years of movies directed by James Cameron that grossed over 100 million dollars, whose stars were born in England</i>	4
2.1	Matching form schemata in the Auto domain. Different labels are used to represent the same concept	8
2.2	Examples of web search interfaces illustrating the variability in form layout design	9
2.3	Label histogram for Auto, Airfare, and Book search forms.....	10
2.4	Components of Web-forms	11
2.5	Prudent schema-matching framework.....	12
2.6	There is no single best way to combine similarity for different domains	16
2.7	No validation can lead to incorrect matchings and consequent errors	19
2.8	Effectiveness of FormMatch versus HSM on TEL8 and FFC1_Man dataset.....	23
2.9	Different combination strategies in MD	24
2.10	PruSM performance with raw and clean data	25
2.11	HSM performance with raw and clean data	26
2.12	The effectiveness of Matching Discovery with different correlation thresholds	27
2.13	Effectiveness of FormMatch on FFC2 dataset with automatic label extraction	28
2.14	Effectiveness of FormMatch on different datasets.....	28
2.15	Adjusted evaluation.....	29
3.1	Excerpts from English and Portuguese infoboxes for the film The Last Emperor	37
3.2	Matching Multilingual Infoboxes	39
3.3	Some attributes for Actor in Pt-En.....	42

3.4	All configurations tried with COMA++	50
3.5	PR-Curves of different approaches in PT	51
3.6	Impact of different thresholds on WikiMatch	55
3.7	Cumulative Gain of k answers	57
4.1	PruSM matching framework	62
4.2	No validation can lead to incorrect matchings and consequent errors	p64
4.3	Contribution of different features in FormMatch	67
4.4	Contribution of different features in WikiMatch	68
4.5	Threshold sensitivity of FormMatch	68
4.6	Impact of different thresholds on WikiMatch	69
5.1	Different template names are used for the type Character	71
5.2	Example of infobox and category names of a Movie	72
5.3	The long tail schema histogram	74
5.4	Illustration of the three-step entity discovery approach	76
5.5	Cluster Reconciliation using link structure and schema similarities	82
5.6	Example of Semantic Refinement	84
5.7	Result of the entity clusters by using different clustering strategies in Movie, Book, and Computer domain	88
5.8	Cluster cohesion based on DBpedia ontology	90
5.9	Precision and Coverage of the attribute clusters when changing Ts, Tg, and the number of ambiguous attributes	95
5.10	Precision and coverage of attribute clusters for different values of Ds and Dg	96
5.11	Excerpt of the Schema Graph in the Movie domain	97
5.12	WikiQuery results: WIClust types vs. Wiki Categories	98

LIST OF TABLES

2.1	Example of input schemata and their attribute pairs	18
2.2	Example of matches discovered in the Matching Discovery step for FFC Auto	19
2.3	Experimental hidden-Web domains	22
2.4	Effectiveness of DCM reduced when not applying preprocessing.....	32
2.5	Effectiveness of HSM and DCM reduced when considering rare attributes.....	33
3.1	Overlapping infoboxes	37
3.2	Weighted Precision (P), Recall (R), and F-measure (F) for the different approaches.....	49
3.3	Results by the classifier with pseudo-training and consistency learning.....	52
3.4	Macro-averaging results	53
3.5	Contribution of different components	54
3.6	MAP for different sources of correlation	54
3.7	List of c-queries used in the case study and their meaning	56
5.1	Heterogeneity of template and category names.....	73
5.2	Constraints used in the Attribute Clustering step	79
5.3	Description and characteristics of the dataset	86
5.4	Accuracy and the confidence interval of WIClust	89
5.5	Entity clusters before and after reconciliation.....	92
5.6	Some synonym attributes identified	92
5.7	Precision and coverage of the attribute clusters in the Movie domain	94

ACKNOWLEDGEMENTS

It would not have been possible to write this doctoral dissertation without the guidance of my committee members, help from friends, and support from my family. It is a pleasure to convey my gratitude to them all in my humble acknowledgment.

First of all, I would like to express my deepest gratitude to my advisor, Prof. Juliana Freire, for her great supervision, support, understanding, and patience during my graduate studies at University of Utah. I thank you for your profound understanding and crucial support during a hard time of my life. Without her support and guidance, I would not have been able to finish this dissertation. I consider myself to be extremely fortunate to have her as my advisor. For everything that you have done for me, I sincerely thank you, Prof. Juliana Freire.

I would like to acknowledge all of my committee members, Prof. Claudio Silva, Prof. Suresh Venkat, Prof. Renée J. Miller, and Dr. Viviane Moreira, for their insightful discussions and constructive comments to improve my dissertation.

I would like to thank my colleagues in the WebDB group at the University of Utah — Luciano Barbosa, Ramesh Pinnameneni, Huong Nguyen, Hoa Nguyen, Viviane Moreira, and Karane Viera — for your friendship and valuable discussions. It has been a great pleasure to collaborate with all of you.

I would like to thank the staff members at the School of Computing (SoC) and Scientific Computing and Imaging Institute (SCI) for their support during my graduate studies at the University of Utah. Many thanks go in particular to Mrs. Karen Feinauer and Mrs. Ann Carlstrom.

With all my heart, I would like to thank my parents and my parents-in-law in Vietnam. They have unconditionally supported me in all my endeavors. I would like to thank them for their sacrifices during my study and hope this dissertation somehow expresses my appreciation to them.

I would like to give special thanks to my husband Tuong Huynh for his personal support, endless love, and persistent patience during my time in graduate school.

Last but not least, I would like to thank my little daughter Kate Huynh, who was born before this dissertation was completed. Thank you my little dear for bringing me a lot of love and encouragement when writing this dissertation.

CHAPTER 1

INTRODUCTION

1.1 Motivation

There has been an explosion in the volume of structured data on the Web. The availability of structured data (Figures 1.1 and 1.2) in online databases, product catalogs, or resources such as Wikipedia, creates new opportunities for querying these data that go far beyond the simple search capabilities provided by search engines. Recognizing this opportunity, several applications have emerged that support complex queries over Web data. One can search for his or her dream car (Figure 1.1), compare different laptop models (Figure 1.3), or pose trivia questions about your favorite movies and celebrities (Figure 1.4). But unlike traditional databases, data on the Web do not come with an explicit schema specifying the attributes and their types. In addition, data published in different sites can differ in structure (e.g., they can have different implicit schemata) and semantics. Consequently, to support complex, structured queries, substantial work is required to extract and integrate the necessary information.

	<u>2005 Acura NSX</u> Berlina Black, 2 door, RWD, Coupe, 6-Speed Manual, 3.2L V6 24V MPFI DOHC, Stock# 05ANBBA. Dealer: Exclusive Motorcars (Los Angeles, CA ~ 2 mi. away) ☎ 877-438-3630 ✉ Email Dealer ✓ Free CARFAX Report	21,992 mi.	\$69,995	<input type="checkbox"/>
	<u>1995 Acura NSX T</u> Red, 2 door, RWD, Coupe, Manual, 6 Cylinders. Individual Seller: JESSICA SUH (LOS ANGELES, CA ~ 2 mi. away) ☎ 213-220-6410 (Daytime) 213-220-6410 (Evening) ✉ Email Seller ✓ Free CARFAX Report	9,288 mi.	\$49,999	<input type="checkbox"/>
	<u>2012 Acura MDX TECHENT</u> Polished Metal Metallic, 5 door, SUV, 6AT, 3.7L, Stock# 14153. Dealer: Valencia Acura (Valencia, CA ~ 27 mi. away) ☎ 866-891-2688 ✓ Free CARFAX Report	4,080 mi.	\$42,599	<input type="checkbox"/>

Figure 1.1: Information aggregators integrate information from multiple sources and provide a unified interface that allows user to query this information. Using Yahoo! Autos, for example, users can search for cars that are advertised all over the United States.

Bay Area Cities

 FILTER BY: Total Population

Name	2010 Total Population	2000 Total Population	2000-2010 Change
Alameda	73812	72259	▲2.1%
Alamo CDP	14570	15626	▼6.8%
Albany	18539	16444	▲12.7%
Alum Rock CDP	15536	13479	▲15.3%
American Canyon	19454	9774	▲99.0%
Angwin CDP	3051	3148	▼3.1%
Antioch	102372	90532	▲13.1%

(a)

Steven Allan Spielberg (born December 18, 1946)^[5] is an American film director, screenwriter, producer, [video game designer](#), and studio entrepreneur. In a career of more than four decades, [Spielberg's films](#) have covered many themes and genres. Spielberg's early science-fiction and adventure films were seen as archetypes of modern Hollywood blockbuster filmmaking. In later years, his films began addressing issues such as [the Holocaust](#), the [Transatlantic slave trade](#), war, and terrorism. He is considered one of the most popular and influential filmmakers in the history of cinema.^[6] He is also one of the co-founders of [DreamWorks](#) movie studio.

Spielberg won the [Academy Award for Best Director](#) for [Schindler's List](#) (1993) and [Saving Private Ryan](#) (1998). Three of Spielberg's films—[Jaws](#) (1975), [E.T. the Extra-Terrestrial](#) (1982), and [Jurassic Park](#) (1993)—achieved [box office](#) records, each becoming the highest-

Steven Spielberg



Steven Spielberg speaking at the [Pentagon](#) on August 11, 1999 after receiving the Department of Defense Medal for Distinguished Public Service from Secretary of Defense [William S. Cohen](#)

Born Steven Allan Spielberg
December 18, 1946
(age 65)^[1]
[Cincinnati](#), Ohio, U.S.

Nationality American

Education [Saratoga High School](#)

Alma mater [California State University,](#)

(b)

Figure 1.2: Structured information on the Web is available not only in the Deep Web, hidden behind Web-form interfaces, but it is also published on the Web surface as tables (a) and records with implicit structure (b).




 <p><u>Asus Eee PC X101-EU17-WT 10.1" LED Netbook - Intel Atom N435 1.33 GHz - White</u></p> <p><u>\$171</u>(4 stores)</p>	 <p><u>Apple MacBook Pro 15.4" - Core 2 Duo - MacOS X 10.6 - 4 GB RAM - 250 GB HDD</u></p> <p><u>\$999</u>(1 store) 4.5 out of 5 <u>Customer reviews</u> (3)</p>	 <p><u>HP ProBook VM614AV 15.6" LED Barebone Notebook - AMD M880G Chipset - Phenom II, Turion II, Athlon II Support</u></p> <p><u>\$116</u>(2 stores)</p>
<p>PROCESSOR & CHIPSET</p> <p>Processor Manufacturer Intel</p> <p>Processor Type Atom</p> <p>Processor Model N435</p> <p>Processor Speed 1.33 GHz</p> <p>Cache 512 KB</p>	<p>PROCESSOR & CHIPSET</p> <p>Chipset Manufacturer AMD</p> <p>Chipset Model M880G</p> <p>Processor Supported Turion II Athlon II Phenom II</p>	<p>PROCESSOR / CHIPSET</p> <p>CPU Intel Core 2 Duo 2.53 GHz</p> <p>Number of Cores Dual-Core</p> <p>Cache L2 - 3 MB</p> <p>64-bit Computing Yes</p> <p>Front Side Bus 1066 MHz</p>

Figure 1.3: To support faceted queries or to compare products advertised by different vendors, product search engines must identify correspondences between the different schemata they use. For example, for laptops, different terms are used to represent the CPU used (Chipset, Processor).

While there is extensive literature on the topic of data integration [88, 91, 55, 107, 52, 53, 93, 102, 106, 5, 38, 41], Web-scale integration tasks bring new challenges. Notably, structured data on the Web are highly heterogeneous and noisy, and since there are many data sources, it is not practical to rely on approaches that require well-defined schemata, clean data, or substantial manual intervention. In this dissertation, we examine the problem of large-scale information integration. We develop techniques and algorithms that automate, to a great extent, the integration of large collections of structured data. More specifically, we address the following problems: schema-matching for Web-forms [81, 80], discovery of entity types and relationships for Wikipedia infoboxes [82], and multilingual schema-matching for Wikipedia infoboxes [79]. Compared to previous approaches

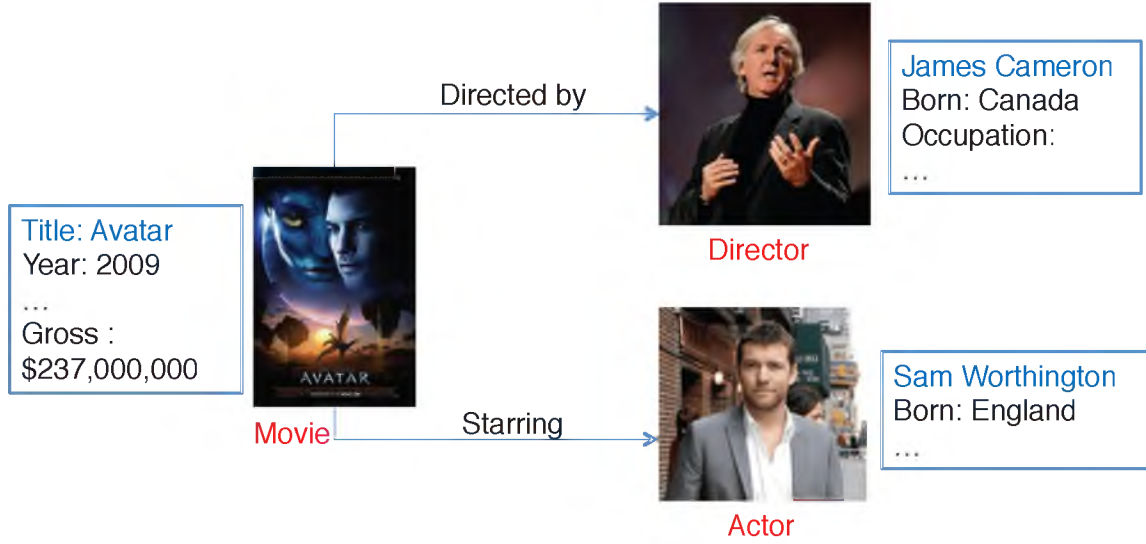


Figure 1.4: The availability of structured information in the form of infoboxes makes it possible to answer complex queries such as *Find the titles and years of movies directed by James Cameron that grossed over 100 million dollars, whose stars were born in England.*

to these problems, an important advantage of the techniques we have developed is that they are automatic and follow a data-driven process that leverages the availability of a large number of data sources to both discover (implicit) structure within the data as well as correspondences across data sources.

1.2 Technical Contributions and Outline

Our main technical contributions are summarized as below:

- **Matching Web-form Schemata (Chapter 2):** We propose `FormMatch`, an effective and scalable approach for matching a large number of Web-form schemata. A form is modeled as a set of elements. Given a set of Web-forms $F = \bigcup_{i=1}^n f_i$ in a given domain D , we aim to identify all the correspondences (matches) among elements across forms $f_i \in F$, and group them into clusters $C_i = \{C_1, C_2, \dots, C_k\}$, where each cluster contains only elements that share the same meaning. `FormMatch` combines multiple sources of similarities in such a way that the different sources reinforce each other. In addition, it prioritizes matches with the highest confidence. It then uses the high-confidence matches to resolve ambiguities and incrementally grow the set of final matches. This two-step process not only avoids error propagation, but it also leads to a higher recall. The results of an extensive experimental evaluation show that `FormMatch` obtains high matching accuracy even in the presence of noise and rare attributes, outperforming other schema-matching approaches [53, 93] which unlike `FormMatch`, require the forms to be preprocessed and attribute labels to be clean by

applying manual preprocessing for the data.

- **Multilingual schema-matching for Wikipedia infoboxes (Chapter 3):** As a step towards supporting multilingual queries over Wikipedia content, we propose `WikiMatch`, a new approach that identifies mappings between attributes from infoboxes that come from pages in different languages. Our approach finds mappings in a completely automated fashion. Because it does not rely on supervised learning techniques, it is scalable: not only can it be used to find mappings between many language pairs, but it is also effective for languages that are underrepresented and lack sufficient training samples. Another important benefit of our approach is that it does not depend on syntactic similarity between attribute names, and thus, it can be applied to language pairs that have distinct morphologies. Similar to our approach to Web-form schema-matching, `WikiMatch` combines multiple sources of similarity to derive matches and it also prioritizes high-confidence matches. We have performed an extensive experimental evaluation using a corpus consisting of pages in Portuguese, Vietnamese, and English. The results show that not only does our approach obtain high precision and recall, but it also outperforms state-of-the-art techniques. We also present a case study which shows that the multilingual mappings we derive lead to substantial improvements in answer quality and coverage for structured queries over Wikipedia content.
- **The PruSM matching framework (Chapter 4):** While designing the techniques to match Web-form interfaces (`FormMatch`) and multilingual schemata (`WikiMatch`), we have identified important features that are effective in the derivation of matches for large-scale Web integration tasks. Although there are multiple sources of similarity, there is no single best way to combine the similarities. In order to have at least one high-precision matcher, we define comprehensive constraints that combine correlations with other similarities to obtain high-confidence matches first and minimize propagation errors. We propose `PruSM`, a prudent matching framework which generalizes both `FormMatch` and `WikiMatch`. We also show how `PruSM` can be customized for different integration scenarios.
- **Organizing Wikipedia infoboxes (Chapter 5):** Wikipedia has emerged as an important source of structured information on the Web. However, while the success of Wikipedia can be attributed in part to the simplicity of adding and modifying content, this has also created challenges when it comes to using, querying, and integrating the information. Even though authors are encouraged to select appropriate categories and provide infoboxes that follow predefined templates, many do not follow the guidelines or follow them loosely. This leads to undesirable effects, such as template duplication and schema heterogeneity. We propose `WIClust`, a new approach that automatically clusters Wikipedia infoboxes to discover entity

types and their relationships. `WIClust` does not require the number of types to be known in advance, it is resilient to the high skew present in the data, and it is robust in the presence of rare and optional attributes. Because the process is automated, it gracefully supports the dynamic nature of Wikipedia, and since it relies only on the structure of the infoboxes, it can be applied to infoboxes in different languages. We perform a detailed experimental evaluation in three distinct domains, using over 107K infoboxes. The results show that our clustering algorithm discovers meaningful entity types and derives high-quality clusters, outperforming other clustering techniques in terms of both F-measure and cohesion. A comparison between the types automatically discovered by our approach against the types manually assigned to infoboxes in DBpedia shows that our approach is effective: not only does it derive clusters that include the manually created types, but it also includes types that are not covered by DBpedia. We present case studies that show how the results derived by `WIClust` are useful to support complex and multilingual queries over Wikipedia.

CHAPTER 2

MATCHING WEB-FORM INTERFACES

2.1 Introduction

It is estimated there are millions of databases on the Web whose contents are hidden and are only exposed on demand, as users fill out and submit Web-forms [69]. Several applications have emerged which attempt to uncover hidden-Web information and make it more easily accessible, including meta-searchers [47, 48], hidden-Web crawlers [12, 70], and Web information integration systems [26, 55]. These applications face several challenges, from locating relevant forms [13], determining their domains [15, 14], and understanding the semantics of the form elements [108, 77]. Consider, for example, meta-searchers and Web information integration systems that provide access to multiple sites of the same domain through a unified query interface. To build this interface, it is necessary to first solve the problem of *Web-form schema-matching*: Given a large set of Web-forms, automatically identify the *correspondences* (or *matches*) among elements in these forms.

While there has been substantial work in the area of schema-matching [88, 91], new challenges emerge for matching form schemata. The schemata are not explicitly defined and need to be extracted from the HTML pages [108, 77], which can lead to errors being introduced. Also, the data are highly heterogeneous—there is a wide variability in how forms are designed, even within a well-defined domain. A number of approaches have been proposed for form schema-matching [107, 52, 93, 102, 106]. However, these are based on the assumption that the input for the matching step consists of clean schemata, which in practice, requires substantial manual intervention both to extract the correct labels from the HTML forms and to normalize these labels (see details below). Consequently, these approaches are not effective for large collections of forms where manual preprocessing is not practical.

To identify matches, an important step is to define how the similarity between attributes and schemata should be computed. There are different sources of similarity that can be considered for Web-forms, including the attribute names and their values, when present. However, combining these different sources of similarity is challenging; no single strategy is uniformly good across different domains or even for pairs of schemata, as Figure 2.1 illustrates. Different labels, including labels with no syntactic similarity (e.g., `make` and `manufacturers`), are used to represent the same concept, while syntactically similar labels (e.g., `manufacturer` and `year of manufacture`) are used

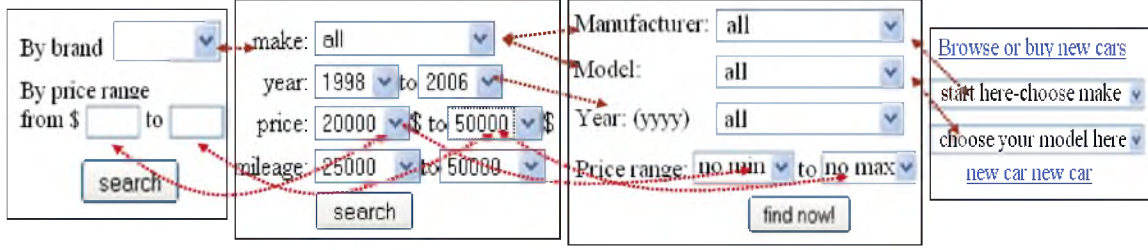


Figure 2.1: Matching form schemata in the Auto domain. Different labels are used to represent the same concept.

to represent different concepts. Besides labels, element values are another source of similarity information that can be used to derive correspondences. However, similar to labels, they can lead to mistakes. For example, `price` and `mileage` have similar values and yet represent different concepts.

In addition, there is significant variability in form layout. For example, as Figure 2.2 illustrates, labels can be placed in many different positions: on top, in the bottom, to the left and to the right of an element, and even inside a form element (e.g., internal values). This leads to difficulties in automatically extracting labels for form elements [77]. As a result, automatic label extraction invariably leads to errors which can negatively impact the matching process. Due to the variability in form design and the errors introduced by label extraction, approaches that employ one individual matcher or a fixed combination of matchers [55, 107, 69, 102] are likely to have low accuracy.

Statistical matching approaches [52, 53, 93] were proposed that take advantage of the availability of a large number of forms. The key intuition behind these approaches is that attribute correlation can provide a reliable source for similarity (and dissimilarity) information. For example, if two attributes have high positive correlation (i.e., they often co-occur together in forms), they should not be *synonyms*. Conversely, if their negative correlation is high, they may be synonyms. The effectiveness of these approaches, however, depends on the availability of *manually* preprocessed, clean form collection. As part of the preprocessing, they apply *syntactic merging* to normalize labels. This includes the removal of stop words and supplement words. For example, `please select a make` and `make` are merged. If done manually, syntactic merging is very labor-intensive. Although label normalization can be automated, Dragut et al. [37] have shown that it can be problematic—even simple stop word removal can lead to mistakes, as the following example shows.

Example 1. There are many linguistic variants for labels representing the concept *Model* in the Auto search forms: `Choose vehicle model` `example mustang`, `If so what is your model`, `Please choose a vehicle`, etc. While *Model* is usually an important term, it is not important in *Model Year*. Stop words can also be deceiving. While `to` is a stop word in `fly to`, it also means *destination* when used in isolation in *airfare* search forms. ■

Figure 2.2: Examples of web search interfaces illustrating the variability in form layout design.

Another limitation of these approaches comes from the fact that while correlation is an effective measure for attributes that have high frequency in a form collection, it fails for rare attributes. Consequently, statistical approaches ignore low-frequency attributes. This can be problematic and result in low coverage for the derived matches, since due to the high variability in the labels used, the distribution of labels follows a Zipf-like distribution—few labels are frequent and many labels are infrequent. We illustrated this in Figure 2.3, which shows the long tail label histogram in different domains and datasets.

To address these challenges, we proposed `FormMatch` [81, 80], a form-matching strategy that is resilient to noise and rare attributes which are commonplace in large form collections. As we describe below, `FormMatch` prioritizes matches with high confidence by incorporating both syntactic and latent information in an aggregated fashion, i.e., considering *sets* of elements. In addition, it combines different similarity sources in a prudent way so that these features reinforce each other. By doing so, `FormMatch` minimizes the propagation of matching errors. Last but not least, `FormMatch` uses the initial, high-confidence matches to resolve uncertain ones, substantially increasing its recall.

We have evaluated `FormMatch` using 4,577 forms from multiple domains. Our experiments show that `FormMatch` obtains high precision and recall without any manual preprocessing, has higher accuracy (between 10% and 68%) than state-of-the-art matching approaches, and it is able to reliably find matches for infrequent attributes.

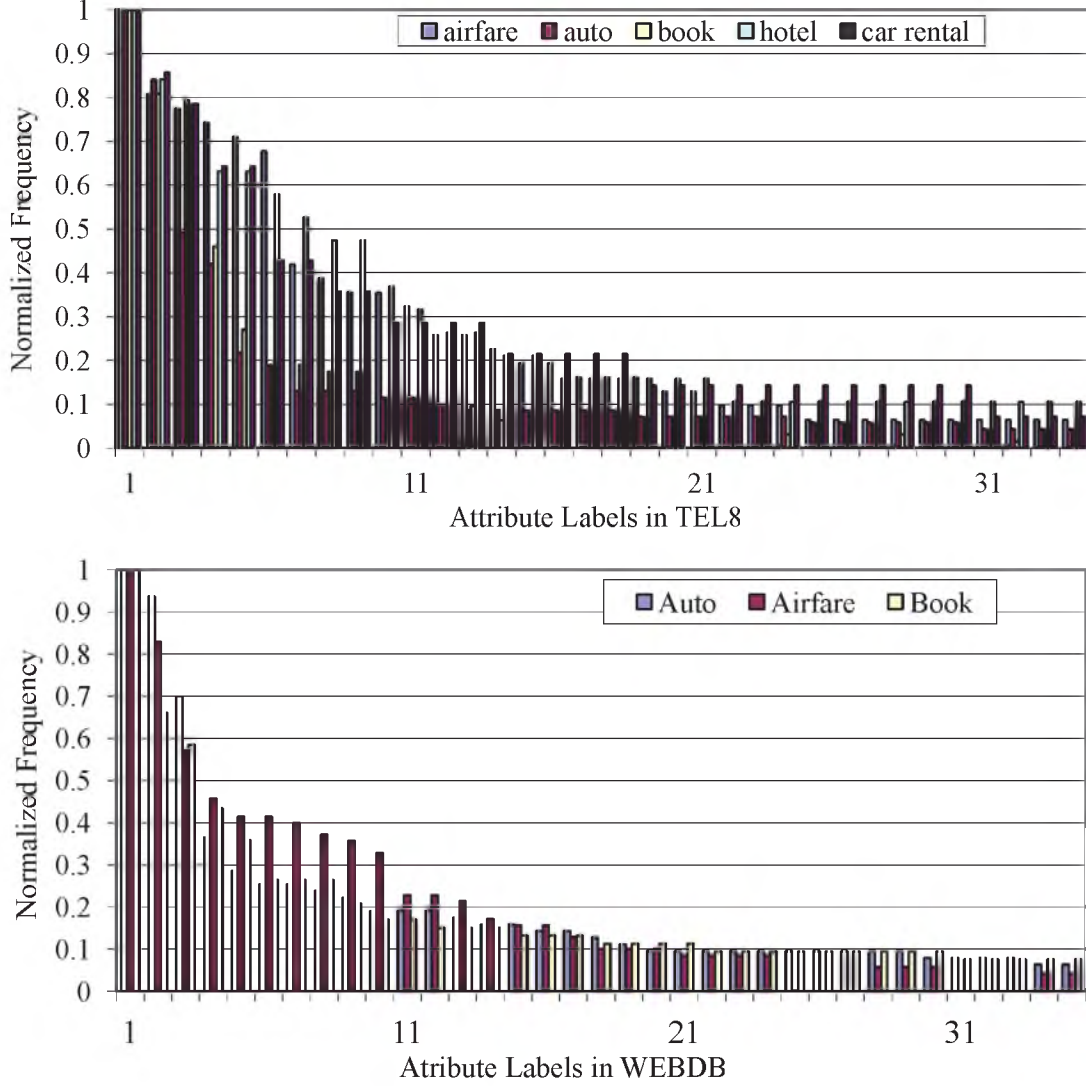


Figure 2.3: Label histogram for Auto, Airfare, and Book search forms

2.2 Problem Definition and Solution Overview

2.2.1 Problem Definition

Before defining the problem, let us take a look at the anatomy of a Web-form in Figure 2.4. A Web-form F contains a set of *elements* $E = \{e_1, e_2, \dots, e_n\}$. Each element e_i is represented by a tuple (l_i, v_i) , where l_i is the label of e_i (i.e., a textual description) and v_i is a vector that contains a list of possible domain values for e_i . Since the term ordering is used differently for different labels (e.g., *vehicle year* versus *year of vehicle*), we consider an element label l as a bag of terms $\{t_1, t_2, \dots, t_m\}$, where each term t_i is associated a weight w_i . For example, there are five elements in Figure 2.4. The element labels of the form are *Make*, *Model*, *Maximum price*, *Search within*, and

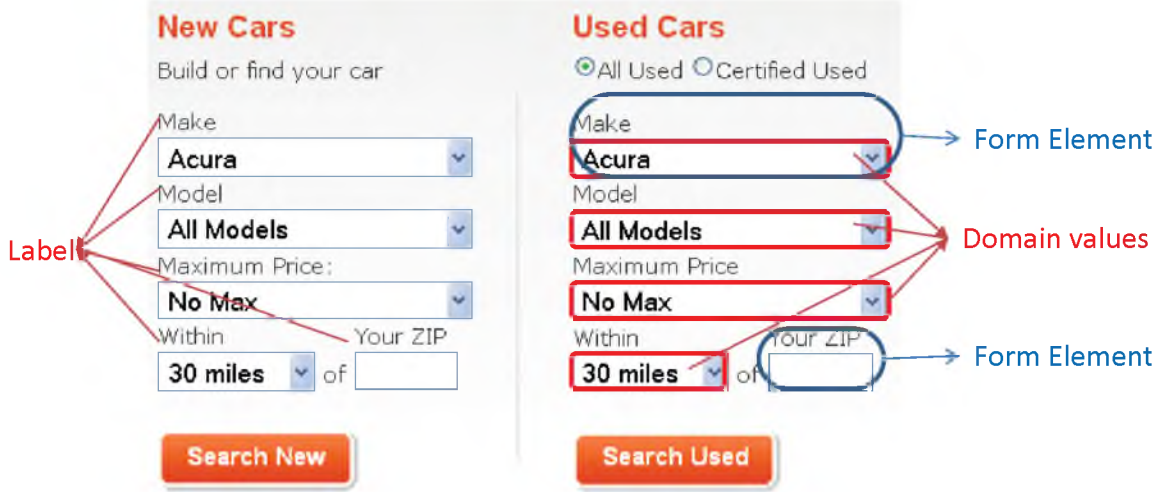


Figure 2.4: Components of Web-forms

Your ZIP; domain values for the element Model are {All, MDX, RDX, RL, TL, TSX}. The composite label Your ZIP consists of two terms, and intuitively, ZIP is the most important term since it conveys the meaning of the element. Thus, it should be associated with a higher weight than with Your.

The *Web-form schema-matching problem* can be stated as follows. Given a set of Web-forms F in a domain D , the schema matching process identifies the set M of all the *matches* among elements across forms. A match m_i consists of a set of attributes or groups of attributes: $m_i = \{g_{i1} \sim \dots \sim g_{iw}\}$, where g is a group of attribute $\{a_k\}$. For 1:1 matches, $k = 1$ and for complex matches, $k > 1$. From M , we derive a set of clusters $C = \{C_1, \dots, C_m\}$, where each cluster contains only elements that share the same or similar meaning.

2.2.2 Solution Overview

The form-matching framework is illustrated in Figure 2.5. It consists of three components: Aggregation, Matching Discovery, and Matching Growth. Given a set of form schemata, the *Aggregation* module groups together similar attributes, divides and sends the set of frequent attributes (S_1) to the *Matching Discovery* module, and the set of infrequent attributes (S_2) to the *Matching Growth* module. Matching Discovery finds matches (both 1:1 and n:m) among frequent attributes. These matches M , together with infrequent attributes S_2 , are used by Matching Growth to obtain additional matches that include infrequent attributes.

By aggregating elements that have the same label, the Aggregation component can improve value distribution and take the benefit of most common and available domain values to reduce domain value sparseness. It is also fundamental for the next steps of FormMatch where measurements are based on a set of elements (attribute correlations and domain-value similarity). Note that

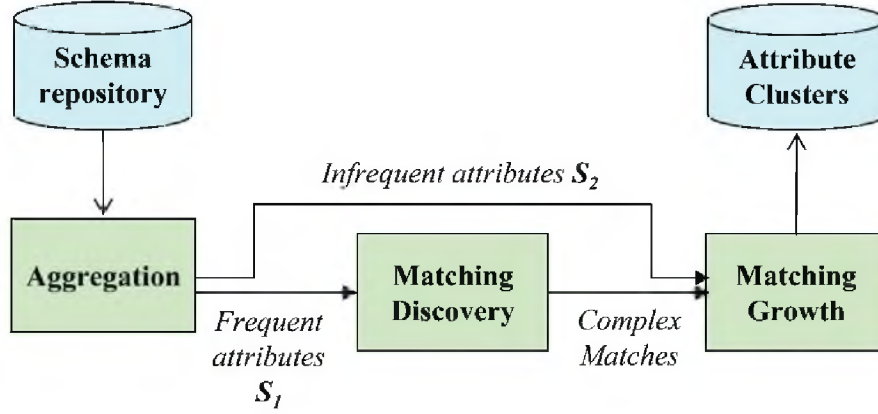


Figure 2.5: Prudent schema-matching framework

previous statistic matching approaches [52, 53, 93] require that forms be preprocessed to remove irrelevant terms and *simplify* the labels. For instance, `Search for book titles` is simplified to `title`. In contrast, FormMatch does not need to detect and remove domain-specific words (e.g., `car`, `vehicle`, `auto`, `books`) or generic search terms (e.g., `select`, `choose`, `find`, `enter`, `please`) or modification terms (e.g., `a range of`, `if so`, `what is your`, ...)

In order to determine matches, FormMatch leverages multiple sources of similarity, including label similarity, value similarity, and attribute correlation (see Section 2.3.1). Because of the heterogeneity of Web-forms, these similarities are combined in a prudent way (Section 2.3.2). After the Matching Discovery step where initial high-confidence matches are derived, FormMatch performs the Matching Growth (MG) which finds additional matches for rare attributes. The FormMatch algorithm is outlined in Algorithm 1.

2.3 Form Matching

2.3.1 Computing Similarities

Instead of computing the similarity for single elements, we compute the similarity for form attributes which are composed of a set of elements. For each pair of attributes (a_i, a_j) , we quantify the similarity between them using three measures: label similarity, domain-value similarity, and correlation. Below we describe these measures, their benefits and limitations and then we present how to combine these measures.

2.3.1.1 Label Similarity

Because forms are designed for human consumption, labels are descriptive and are often an important source for identifying similar elements. White space tokenizing serves as a good delimiter

Algorithm 1 Form Matching

```

1: Input: Set of attributes  $A$  present in a set of forms  $F$  in domain  $D$ , configuration  $Conf$ , grouping
   threshold  $T_g$ 
2: Output: Set of attribute clusters  $\{C_1, \dots, C_n\}$  corresponding to the identified matches
3: begin
4: /*1. Aggregation*/
5:   Stem terms in labels
6:   Aggregate elements that have the same label
7:   Create set  $S_1$  of frequent and  $S_2$  of infrequent attributes
8: /*2. Matching Discovery for frequent attributes*/
9: /* Compute the similarities for each pair of attributes */
10:  $P = \{ \langle a_p, a_q \rangle, lsim_{pq}, dsim_{pq}, X_{pq}, Y_{pq} \} | a_p, a_q \in S_1 \}$ 
11:  $M \leftarrow \emptyset$ 
12: while  $P \neq \emptyset$  do
13:   Choose attribute pair  $\langle a_p, a_q \rangle$  that have the highest  $X_{pq}$ 
14:   if  $validation(a_p, a_q), Conf$  then
15:      $M \leftarrow \text{IntegrateMatches}(a_p, a_q, M, T_g)$ 
16:   else
17:     /*buffering uncertain matches*/
18:      $B \leftarrow \langle a_p, a_q \rangle$ 
19:   end if
20:   Remove  $\langle a_p, a_q \rangle$  from  $P$ 
21: end while
22: /* Resolve uncertain matches in buffer  $B$  using IntegrateMatches */
23:  $M \leftarrow \text{IntegrateMatches}(B, M)$ 
24: /*3. Matching Growth for rare attributes*/
25:   Create a set of clusters  $\{C\}$  according to  $M$ 
26:   Update  $STF$  and compute new term weights
27:   Use INN to assign rare attributes to the cluster of closest match
28:   Cluster unmatched attributes by HAC and add them to  $\{C\}$ 

```

in this case to extract terms from the form labels. Grams or camel-case could potentially be helpful for form element names. However, we do not use element names because often, they are not very descriptive and can be null. Most other form-matching methods also consider only the form labels [52, 53, 93, 86].

Each term t_i in the label has a different term weight w_i . We define the label similarity between two attributes (a_i, a_j) as the cosine distance [10] between the *term vectors* for their labels:

$$lsim(a_i, a_j) = \cos(w_i, w_j) \quad (2.1)$$

where

$$\cos(x, y) = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}} \quad (2.2)$$

To capture the importance of terms, in addition to term frequency (TF), we also use the singular token frequency (STF). The intuition behind STF comes from the fact that generic terms such as "Please", "Select", "of", "available" usually have high frequency, since they appear in

many composite labels (e.g., "select" appears in "Select a car make", "Select State", "Select a model") but rarely appear alone. We use STF to distinguish between labels that frequently appear alone and are thus likely to be important, and labels that only appear together with other labels—they do not have a complete meaning by themselves and are thus unlikely to represent an important concept. The term weight w_i is computed in the equations below where TF and STF are computed across the bag of all forms:

$$w(t_i) = \sqrt{TF(t_i) * STF(t_i)} \quad (2.3)$$

$$TF(t_i) = \frac{freq(t_i)}{\sum_{i=1}^n freq(t_i)} \quad (2.4)$$

$$STF(t_i) = \frac{freq(t_i_appear_alone)}{freq(t_i)} \quad (2.5)$$

2.3.1.2 Domain-Value Similarity

To compute the domain similarity between two attributes, we first aggregate all the domain values for each attribute. Given an attribute a_l , we build a vector that contains all occurrences of values associated with the label l for a_l and their frequencies: $V_l = \bigcup_{i=1..n} (v_i : frequency)$. Given two attributes a_i and a_j , the cosine distance (Equation 2.2) is then used to measure the similarity between their corresponding value vectors:

$$dsim(a_i, a_j) = cos(V_i, V_j) \quad (2.6)$$

Example 2. The aggregated values associated with attribute `Make` and `Manufacture` are, respectively: $V_1 = \{\text{make:5, Honda:120, Toyota:150, Camry: 4}\}$, $V_2 = \{\text{Honda:100, Toyota:50, manufacture:2}\}$ ¹. The similarity between them is $dsim(a_1, a_2) = cos(V_1, V_2) = 0.9$. ■

Domain values are a good source of similarity information. However, they are not always available and their domain vector can be empty (e.g., the elements `From` or `To` in Figure 2.1 or `Your zip` in Figure 2.4). Therefore, we consider them as supporting information to validate a match.

2.3.1.3 Correlation

By holistically and simultaneously analyzing a set of forms, we can leverage an implicit source of similarity information: attribute correlation. Correlation is a statistical measure that indicates the strength and direction of a linear relationship between two variables. For Web-forms, we exploit the fact that synonym attributes are semantic alternatives and rarely co-occur in the same

¹make and manufacture are form labels which appear as the internal values in the selection list while Camry, which is a car model, was an error resulting from the label extraction process

form interface (push away)—they are negatively correlated (e.g., *Make* and *Brand*). On the other hand, grouping attributes are semantic complements and often co-occur in the same form interfaces (pull together)—they are positively correlated (e.g., *First name* and *Last name*, *Departure date* and *Return date*). In the context of Web-forms, two correlation measures have been proposed: H-measure [53] and X/Y measure [93]. For FormMatch, we use the latter, which was shown to be better for Web-forms [93] and is defined as follows:

$$X(a_p, a_q) = \begin{cases} 0 & \text{if } a_p, a_q \subset \text{form } f \\ \frac{(C_p - C_{pq})(C_q - C_{pq})}{(C_p + C_q)} & \text{otherwise} \end{cases} \quad (2.7)$$

$$Y(a_p, a_q) = \frac{C_{pq}}{\min(C_p, C_q)} \quad (2.8)$$

where C_p , C_q , and C_{pq} correspond to the number of schemata that contain attribute a_p , a_q , and both of them, respectively. The matching score X captures negative correlation while the grouping score Y captures positive correlation. Intuitively, X is high when a_p, a_q rarely co-occur in the same schema (C_{pq} is small compared to C_p and C_q). In contrast, Y is high when a_p, a_q often co-occur in the same schemata, (C_{pq} is close to C_p and C_q). We note that correlation is not always an accurate measure, in particular, when insufficient instances (e.g., forms containing a specific attribute label) are available.

2.3.2 Matching Discovery

2.3.2.1 Combining the Similarities

Although there are many similarity features, we argue that there is no single best way to combine similarity for different domains, or even pairs of schemata, as illustrated in the following example.

Example 3. As shown in Figure 2.6, *mileage* and *price* sometimes have a similar value range, using domain values can result in an incorrect matching between attributes. On the other hand, if only label similarity is considered, an incorrect match can be derived for *model* and *model year* because they share a term that is important in the collection. Co-occurrence statistics can also be useful to identify mappings; for example, by observing that *make* and *manufacturer* co-occur with a similar set of attributes but rarely co-occur together, it is possible to infer that they are synonym attributes. However, when used in isolation, attribute correlation can lead to an incorrect match between *make* with many other rare attributes like *Budget*, *Original listing price range*. In particular, correlation matching scores can be artificially high for rare attributes that are commonplace for Web-forms, since rare attributes seldom co-occur with (all) other attributes. ■

Figure 2.6: There is no single best way to combine similarity for different domains

Combining these similarities to have a uniformly good result is hard. There is a great variability in how forms are designed: several variations of labels or even elements with no apparent similarity (i.e., they are synonym attributes) are used to represent the same concept while elements with similar labels or values might be different, which makes it difficult to identify the correspondences. Further, the importance of these features varies not only from form to form, but also from domain to domain. Additionally, although *dsim* is often effective, many of the elements do not have any associated domain values e.g., zip, departure from, isbn (72% of the elements in Book domain do not have any associated domain values). Thus, using a fixed combination is ineffective.

2.3.2.2 Prudent Matcher

We propose a systematic method that uses high-level rules to combine different similarity measures. This is important for the Matching Discovery (MD) step, where we want to prioritize matches with high confidence first to minimize the propagation of matching errors. Later, this set of matches is extended incrementally.

We combine the above similarities using a prudent matcher. For Web-forms, a prudent matcher is a configuration that consists of a set of constraints over correlation score, label similarity, and value similarity. We perform match validation: a match is *valid* if $X(a_p, a_q) > T_{Matching_score}$ AND $[dsim(a_p, a_q) > T_{dsim}$ OR $lsim(a_p, a_q) > T_{lsim}]$ (line 14, Algorithm 1). The goal of Prudent Matcher is to identify high-confidence matches. We define singular matchers as corresponding to the above features: label similarity, domain-value similarity, and correlation. Although these features are super-features, i.e., features obtained at the level of sets of elements, using them in isolation is insufficient and leads to error propagation. However, we observe that correlation can be effective if prudently combined and reinforced with additional evidence, such as strong label or value similarity. The combination rule ensures that even if two attributes a_p and a_q have a high correlation score, a match will be derived only if additional evidence is available. The prudent matcher is simple yet powerful because it can incorporate both visible and latent information at a high level of a *set* of elements. The thresholds can be manually set or learned. This combination of constraints helps filter

Algorithm 2 IntegrateMatches

```

1: Input: a candidate attribute pair or list of candidate attribute pairs  $\langle a_p, a_q \rangle$ , current matches  $M$ ,
   grouping threshold  $T_g$ 
2: Output: updated set of matches  $M$ 
3: begin
4: if neither  $a_p$  nor  $a_q$  appears  $\in M$  then then
5:    $M \leftarrow M + \{a_p \sim a_q\}$ 
6: else if only one of  $a_p$  and  $a_q$  appears in  $M$  then
7:   /* suppose  $a_p$  appears in  $m_j$  and  $a_q$  does not */
8:   if For each  $a_i \in m_j$ , s.t.  $X_{qi} > 0$  then
9:      $m_j \leftarrow m_j + (\sim \{a_q\})$ 
10:  else if  $\exists g_{jk} \in m_j$  s.t.  $X_{qx} > 0 \forall a_x \in g_{jx, x \neq k}$ 
    and  $Y_{qk} > T_g \forall a_k \in g_{jk}$  then
11:     $g_{jk} \leftarrow g_{jk} + \{a_q\}$ 
12:  end if
13: end if
14: end

```

out many negative errors and minimize their propagation errors. An alternative for this combination is a weighted summation, which is not sufficient for Web-forms. Although domain value similarity itself is often sufficient, many attributes do not have any domain values associated with them. In such a case, the domain similarity is zero does not mean they are different. As shown later, we have the experiments that study the problem of threshold stability. The prudent matcher and the prudent matching framework will be generalized in Chapter 4.

2.3.2.3 Integrate Matches

Only prudent matches are considered to be integrated by *IntegrateMatches* to construct a set of confident matches $M = \{m_j\}$, where each m_j comprises a set of grouping attributes i.e., $m_j = \{g_{j1} \sim \dots \sim g_{jw}\}$. In Algorithm 2, by iteratively considering the highest negatively correlated attribute pairs, it decides whether the attributes will originate a new match, be ignored, or be integrated into an existing match as a new match element or a group element. If attribute a_p and a_q do not exist in M , they are considered to be a completely new match (line 5). If either of them appears in an existing match m_j , the remainder will be checked to become a new match element or a group element in m_j . The idea is to test the negative correlations between all attributes of a match to see whether it is possible to integrate the attributes in question into the existing matches as a new match element (line 9). If attributes that match to the same set of other attributes have a higher positive correlation than T_g , they will become a group element in a match (line 11). A nice property of *IntegrateMatches* is that it gradually merges negatively-correlated attributes together and groups positively-correlated attributes while, at the same time, it pushes away non-negatively-correlated ones. The algorithm is illustrated in the example below:

Example 4. Consider the set of input schemata and their associated frequency F in Table 2.1. Table 2.1 shows attribute pairs that have high matching scores (the normalized matching score is greater than 0.2). The column *Note* shows the outcome of the prudent matcher: F stands for Fail (fail the prudent test), P for Pass, and B for Buffering (uncertain match). Initially, $m_0 = \{\text{make/model} \sim \text{make}\}$. By iteratively integrating confident matches, i.e., the P pairs (pair number 2, 3, 4, 6, 8, 10, 11, 14 and 19), the derived matching result includes $m_0 = \{\text{make/model} \sim \langle \text{make}, \text{model} \rangle \sim \langle \text{select make}, \text{select model} \rangle\}$, $m_1 = \{\text{price} \sim \text{vehicle price}\}$, $m_2 = \{\text{zip} \sim \text{zip code}\}$, $m_3 = \{\text{within} \sim \text{distance}\}$. ■

Example 5. Given the schemata as in Table 2.1, because $X(\text{Make}, \text{Distance}) > X(\text{Within}, \text{Distance})$, the incorrect correspondence between *Make* and *Distance* (the first pair) will eliminate the chance of matching *Distance* and *Within* (the 19th pair) because there is no correspondence between *Make* and *Within* (they co-occur in the interface *S5*). The next incorrect pair is the 5th pair between (*Select make*, *price*). Because *Make* is not matched with *Price* (co-occur in *S2*), the connection of the 6th pair between *Make* and *Select make* cannot be established. ■

We note that, by identifying high confident matches, *FormMatch* can avoid a potentially large number of incorrect matches and its consequent errors. As illustrated in Example 5 and Figure 2.7, using pairs that fail to perform validation (line 14, Algorithm1) can lead to incorrect decisions: if a *bad* decision is made in an early step, it will not be corrected and negatively effect the following steps.

Table 2.1: Example of input schemata and their attribute pairs

#	Schema			F
1	make/model(mm);zip;distance(dis);price(pr)			15
2	make(mk); model(md); price			20
3	select make(sm);model;distance;zip;vehicle price(vpr)			6
4	select make;select model(smd);distance;zip;vehicle price			4
5	make, within, zip code(zc), vehicle price			3
6	make; select model; zip			2

#	X	Pair	Note	#	X	Pair	Note
1	12.5	mk_dis	F	11	4.28	mm_smd	P
2	9.51	mm_md	P	12	2.76	zc_pr	B
3	9.47	pr_vpr	P	13	2.76	within_pr	F
4	9.37	mm_mk	P	14	2.7	zip_zc	P
5	7.77	smk_pr	F	15	2.7	zip_within	B
6	7.14	smk_mk	P	16	2.68	zc_md	B
7	6.96	mm_vpr	F	17	2.68	within_md	F
8	6	mm_smk	P	18	2.67	zc_dis	B
9	5.12	pr_smd	F	19	2.67	within_dis	P
10	4.87	md_smd	P	20	2.5	zc_mm	B

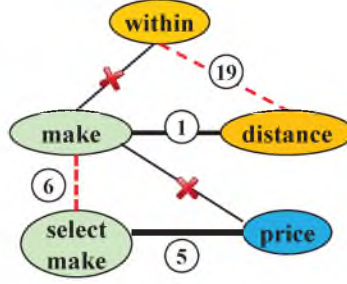


Figure 2.7: No validation can lead to incorrect matchings and consequent errors

The *IntegrateMatches* algorithm is adopted from [93] but using the prudent constraints to avoid error propagation. This algorithm is also similar to the Center clustering algorithm [51] where the attribute pairs are sorted by the similarity scores and are scanned linearly. If an attribute is not covered, this attribute and other uncovered attributes that have similarity higher than a threshold will be added as a new cluster. The difference between the *IntegrateMatches* and Center algorithm is that *IntegrateMatches* is more prudent by requiring every attribute pair inside a cluster to satisfy the constraints. In contrast, Center algorithm is looser and more sensitive to the ordering. Since the validation cannot be totally guaranteed, imperfect ordering and looser constraints may lead to propagation errors, especially when the attribute under consideration is a rare attribute.

The last step in MD is to resolve uncertain but potential matches in the buffer B (line 23, Algorithm 1). By buffering and revising uncertain matches, we can take the advantage of having extra constraints from certain matches to reconcile less certain matches. We apply the *IntegrateMatch* algorithm again to resolve each uncertain pair. It is worth performing this relaxation to consider less certain matches because the domain values sometimes are too coarse, contain no values, or cannot be extracted correctly, leading to a low similarity between them.

Table 2.2 illustrates matches discovered in the Matching Discovery step for the FFC Auto domain. These matches will serve as basic seeds for the Matching Growth step.

Table 2.2: Example of matches discovered in the Matching Discovery step for FFC Auto

select make model ~ make, model ~ select model, select make
year ~ select year ~ year range ~ select rang of model year
price ~ price rang ~ price rang is
zip ~ zip code
valid email ~ email
type ~ body style
search within ~ distance
mile ~ mileage

2.3.3 Matching Growth

After the Matching Discovery step where initial high-confidence matches are derived, `FormMatch` performs the Matching Growth (MG) step, which finds additional matches for rare attributes.

First of all, based on certain matches discovered in the MD phase, the algorithm first updates the STF frequency values to obtain more accurate weights for important terms. Identifying anchor terms that are representative of a domain (e.g., "year", "make", "model" for the auto domain) is very helpful for Matching Growth, where greater variability is present in attribute labels.

Example 6. Assuming we discovered the following match in the Matching Discovery step: `year(44) ~ select year(15) ~ year range(16)`, we can infer that this match contains two supplement terms `select` and `range`. Using this discovered match, we can update the weight of the term `year` and downgrade the weight of the terms `select` and `range` by updating the frequency of the term `year` from 44 to 75. ■

To assign different rare attributes to the cluster of the most similar frequent attributes that we discovered, we use 1-Nearest-Neighbor Clustering [24]. By exploiting the form context and checking the list of matched and unmatched elements of each form, we ensure that two elements in the same form cannot be in the same cluster (namely, the co-location constraint). Examples of additional matches derived by 1NN in Auto domain include `price up to ~ price, price range in euro ~ price range, model example mustang ~ model, approximate mileage ~ mileage, color of corvett ~ color, if so, what's your trade year ~ year`.

Without preprocessing, attribute fragmentation can affect the quality of the correlation and make the correlation scores between attributes lower. In particular, attribute fragmentation happens when attributes co-occur with different sets of attributes that belong to the same concept.

Example 7. Let S be a small set of schemas $S = \{\{A, C_1\}, \{A, C_1, D\}, \{B_1, C_2\}, \{B_1\}, \{B_2, C_1\}\}$. Assuming attributes B_1 and B_2 belong to concept B , C_1 and C_2 belong to concept C , the matching score of A and B_1 is consequently lower than the matching score of A and B . For example, $X(A, B) = 1.2$ while $X(A, B_1) = 1$. ■

Because of validation, we can afford to use a low matching score. However, attribute fragmentation can affect the quality of correlation, leading to incorrect ordering. For example, we encountered the following scenario in Airfare domain: The correlation score $X(\text{departure date}, \text{return on})$ is greater than $X(\text{departure date}, \text{leave on})$. In this case, domain values do not help because they are similar—they all contain values corresponding to months and days. To address this problem, we use attribute proximity information to break ties and find a finer resolution for complex matches. In

this case, the match $\{\text{departure date, return date}\} \sim \{\text{return, depart}\} \sim \{\text{return on, leave on}\}$ can be re-ordered as $\{\text{departure date, return date}\} \sim \{\text{depart, return}\} \sim \{\text{leave on, return on}\}$, and then be broken into clusters of $\{\text{departure date, depart, leave on}\}$ and $\{\text{return date, return, return on}\}$.

Finally, for the remaining unmatched attributes, we run a HAC algorithm (Hierarchical Agglomerative Clustering) [72] to group similar attributes into new clusters and add them to the set of matches. For example, HAC derives and adds the following new clusters: $\{\text{within one month, within one week, within hour}\}$, $\{\text{dealer list, dealer name, omit dealer list}\}$, etc.

The Matching Growth is summarized in Algorithm 1 from lines 24 to 28. As shown in the experimental evaluation, Matching Growth helps improve the final recall significantly.

2.4 Experimental Evaluation

2.4.1 dataset and Evaluation Metrics

2.4.1.1 dataset

We have evaluated `FormMatch` using two public datasets that consist of Web-forms in multiple domains. One of the datasets was manually created and curated; thus, it is small and clean (TEL8²), while the other is a large, heterogeneous collection of forms automatically gathered by a focused crawler [13] and automatically classified into different domains (FFC³). As shown in Figure 2.3, there is a wide variability in the frequency distribution of these labels. In particular, there is a large number of rare attributes, especially in the longer and lower tail of FFC. The labels in TEL8 dataset were manually extracted. The labels in the FFC dataset were extracted both manually and automatically. In particular, FFC2 is the bigger dataset where labels were automatically extracted by a label extraction program [77]. Table 2.3(b) shows information about the dataset, including number of forms, number of form elements, and number of attributes (e.g., sets of element with the same name). We also show the number of elements that do not have any values associated with them. Notably, 72% of elements in the Book domains have an empty value. The last row shows the estimated extraction accuracy in different domains.

In the experiment, we compare `FormMatch` against previous form-matching approaches [53, 93]. Since HSM [93] outperformed DCM [53], we compare `FormMatch` against HSM over the TEL8 and FFC datasets, on 4,577 Web-forms with 33,061 form elements. `FormMatch` does not require manual preprocessing, and it is also impossible to apply syntactic merging in real datasets with a large number of Web-forms. Without syntactic merging, the attributes are often fragmented and

²<http://metaquerier.cs.uiuc.edu/repository>

³<http://www.deeppeep.org>

Table 2.3: Experimental hidden-Web domains

(a) TEL8

Domain	# Forms	# Elements
Auto	98	533
Airfare	49	334
Book	68	403
Hotel	46	281
CarRental	25	191

(b) FFC

Domain	FFC1			FFC2		
	Auto	Airfare	Book	Auto2	Airfare2	Book2
# Forms	136	66	109	2150	851	1290
# Elements	811	745	997	10944	10220	10155
# Attributes	290	162	484	1863	918	2698
% Non-DV Ele	0.39	0.37	0.72	0.44	0.34	0.72
LX Accuracy	1	1	1	0.90	0.84	0.88

thus, in order to have a better coverage for the attributes, we used a low frequency threshold $T_c=5\%$. We reimplement HSM and used the labels as they are—no syntactic merging was applied.

2.4.1.2 Evaluation Metrics

To evaluate the effectiveness of PruSM, we use precision, recall, and F-measure. Precision can be seen as a measure of fidelity, whereas recall is a measure of completeness. F-measure is the harmonic mean between precision and recall—a perfect F-measure has value 1. Given a match m_j , which corresponds to a class i (a class corresponds to a concept), precision and recall are defined as:

$$Pr(m_j) = \frac{n_{ij}}{n_j} \quad (2.9)$$

$$Rc(m_j) = \frac{n_{ij}}{n_i} \quad (2.10)$$

where n_{ij} is the number of elements of class i in match m_j , n_j is the number of elements in m_j , and n_i is the number of members of class i . As there are many matches, we measure the average precision, recall, and F-measure according to the sizes of each match.

$$Pr(M) = \sum_{m_i} \frac{|m_i|}{|M|} * Pr(m_i) \quad (2.11)$$

$$Rc(M) = \sum_{m_i} \frac{|m_i|}{|M|} * Rc(m_i) \quad (2.12)$$

$$F - measure(M) = \frac{2 * Pr(M) * Rc(M)}{Pr(M) + Rc(M)} \quad (2.13)$$

where $|M|$ is the total number of elements present in all matches.

In addition to F-measure, for each match, we compute the cluster entropy according to the probability p_{ij} that a member of cluster j belongs to class i .

$$Entropy(m_j) = -\sum_i p_{ij} * \log p_{ij} \quad (2.14)$$

The total entropy is the sum of the entropy values for all clusters, weighted by the size of each cluster. Intuitively, the more homogeneous are the clusters, the lower is the entropy.

2.4.2 Effectiveness of FormMatch

Figure 2.8 shows the resulting accuracy values of HSM and FormMatch where the FormMatch matching process is split into MD and MG, to show the accuracy obtained in the different phases. HSM has lower accuracy because we use the labels as they are. In particular, HSM has low accuracy in heterogeneous domains like FFC-Book and higher accuracy in ‘clean’ domains like TEL8-Book. For both datasets, and in all domains, the precision, recall, and F-measure values of FormMatch are higher than those for HSM. The gains in F-measure of FormMatch compared to HSM vary between 10% and 39% in TEL8, and between 38% and 68% in FFC1. Smaller improvements are observed in clean domains, where HSM is expected to perform well. While the gains in precision can be attributed to the prudent matching process, the gains in recall come mostly from the Matching Growth phase, which finds additional matches for rare attributes.

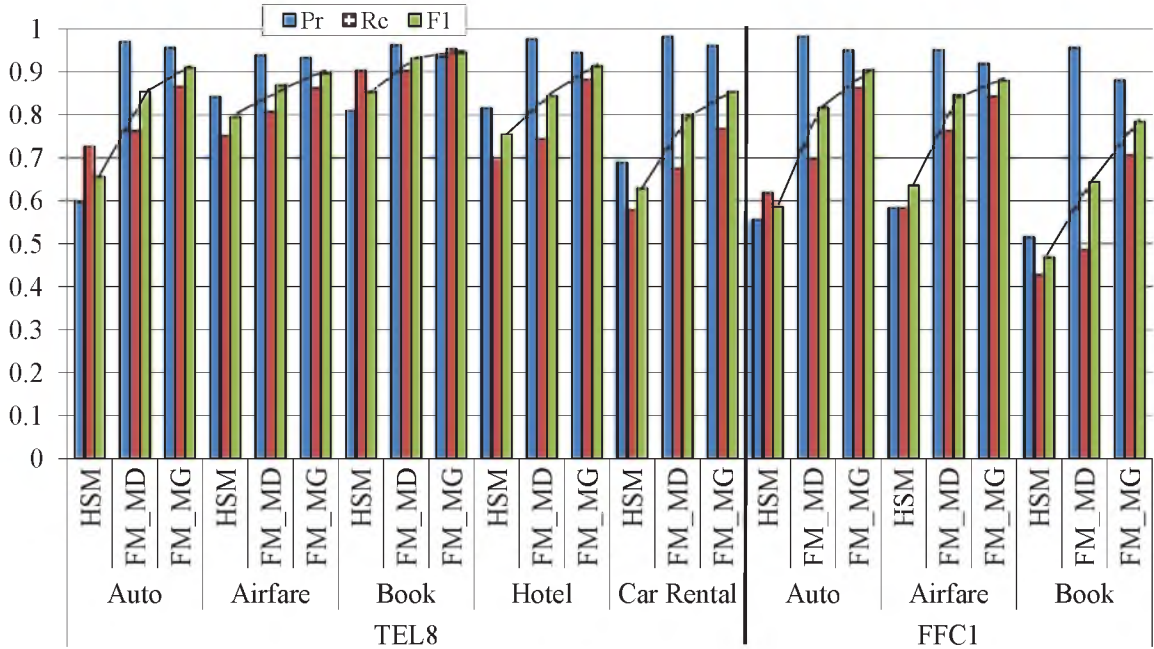


Figure 2.8: Effectiveness of FormMatch versus HSM on TEL8 and FFC1_Man dataset

2.4.2.1 Comparing Different Combinations

We evaluate the effectiveness of the prudent matcher by comparing it against other matchers, including individual matchers (correlation, lsim, dsim), combinations of different matchers like a linear combination of lsim and dsim (Avg2) or lsim, dsim, and correlation (Avg3), or the maximum value among them (Max2, Max3). Figure 2.9 shows that the prudent matcher obtains substantially higher precision than the others, which is the most important goal in MD that we want to obtain. Another possible comparison is to learn the matcher combination, which we leave as future work.

2.4.2.2 Syntactic Merging

We followed the strategy adopted in [53] and manually created a list with generic search-related terms that are present in labels (e.g., “search”, “enter”, “page”, etc.), as well as domain-specific ones (e.g., “book”, “movie”, “vehicle”, etc.). We then ran `FormMatch` and HSM on two variations of the datasets: the original version (raw), and a syntactic merging version where the terms in the list were removed from the element labels (clean). While there was no significant change in F-measure and entropy for `FormMatch` in the presence or absence of these words (Figure 2.10), there was an increase in F-measure and decrease in entropy for HSM on the clean data (Figure 2.11).

2.4.3 Threshold Sensitivity

Besides leading to high-accuracy matches, the prudent matcher also makes `FormMatch` robust to a variation of the correlation threshold. Figure 2.12 shows the sensitivity of the Matching Discovery to different correlation thresholds, obtained by using the correlation matcher e.g., without validation (Figure 2.12(a)) and by using the prudent matcher e.g., with validation (Figure 2.12(b)). As we observe, when the (normalized) correlation threshold increases, the precision increases but recall decreases. When the correlation threshold is lower (from 0.05 to 0.35), the precision is very low

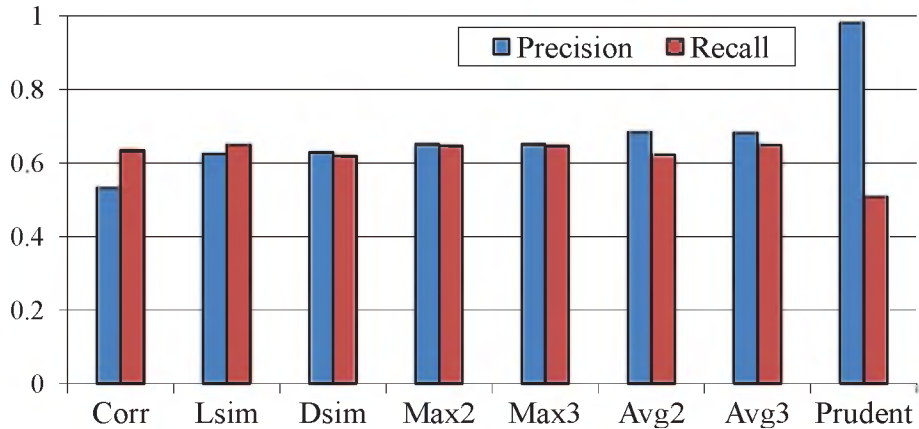
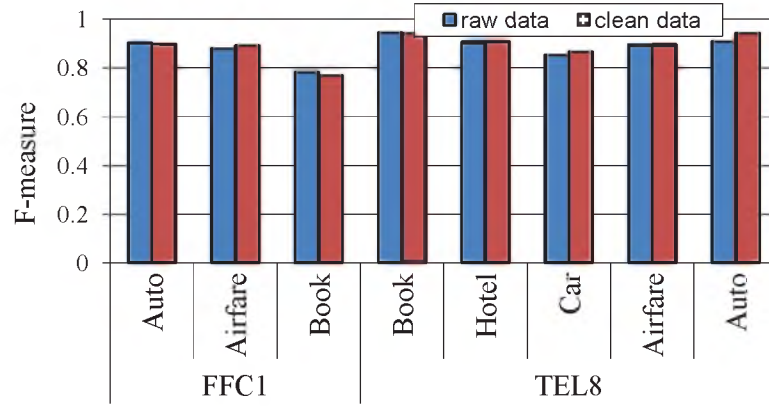
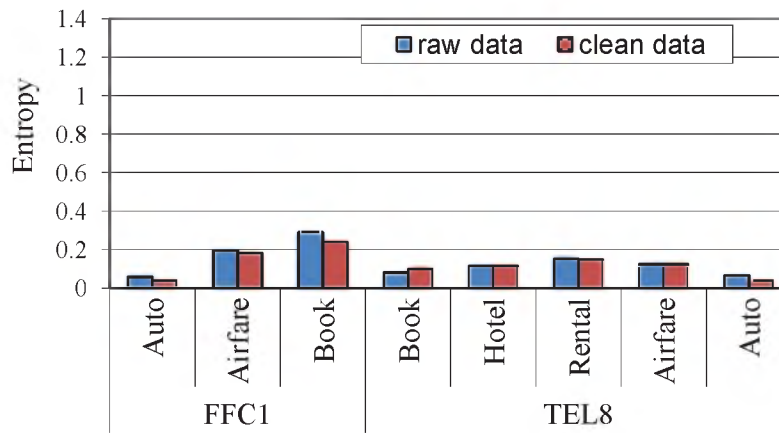


Figure 2.9: Different combination strategies in MD



(a) F-measure



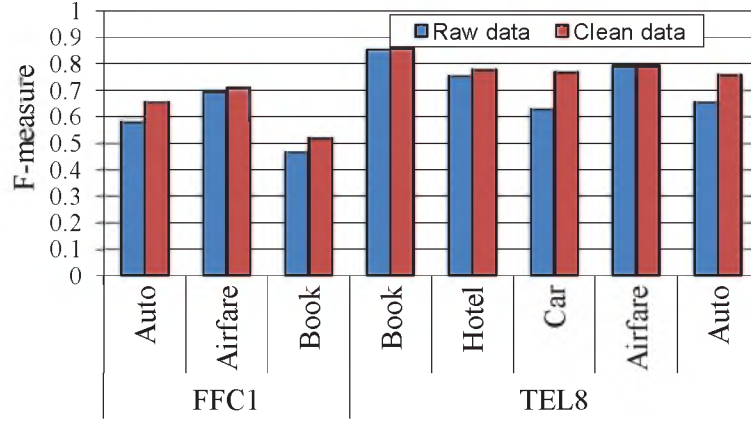
(b) Entropy

Figure 2.10: PruSM performance with raw and clean data

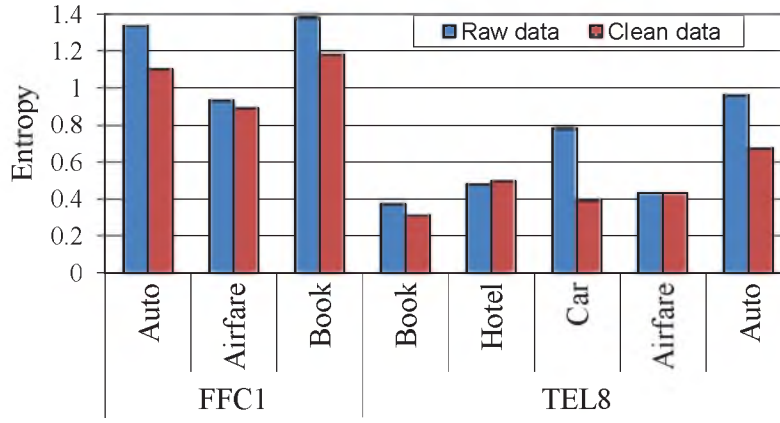
(less than 60%). To obtain an acceptable precision, the correlation threshold must be very high (greater than 0.5), but this also leads to a substantial reduction in recall. On the other hand, when the prudent matcher is used (Figure 2.12(b)), the precision is high even with a very low and wide variation of correlation threshold (from 0.05 to 0.55) and therefore, we can obtain decent values of recall. With decent ranges of label similarity threshold and value similarity threshold, the prudent matcher is very effective and robust to a wide range of correlation threshold. In the experiment of FormMatch, we chose label similarity threshold 0.75 and value similarity threshold 0.5.

2.4.4 Large Dataset with Automatic Label Extraction

Figure 2.13 shows the effectiveness of FormMatch on the large dataset with automatic label extraction (*FFC2_LX*). HSM has lower accuracy because we use the labels as they are and in the case of nonpreprocessing, the correlation signal by itself can mislead the matching process and lead to propagation errors. On the other hand, the prudent matcher helps avoid a large number



(a) F-measure

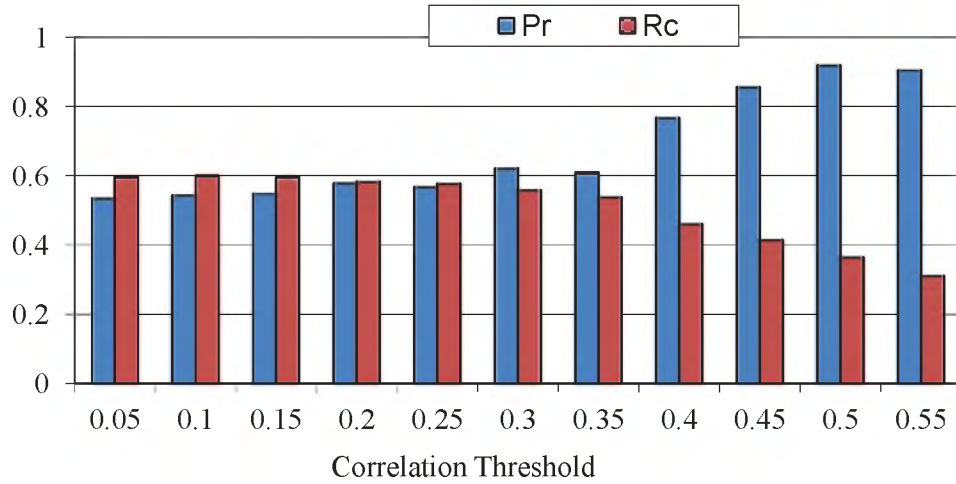


(b) Entropy

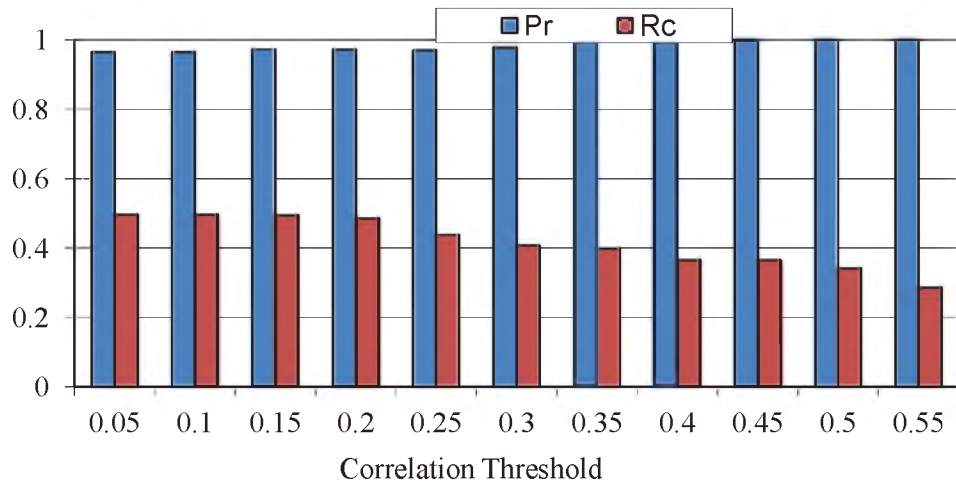
Figure 2.11: HSM performance with raw and clean data

of incorrect matches and its subsequent propagation to obtain high precision, a key requirement for Matching Discovery. The *FormMatch* matching process is split into Matching Discovery and Matching Growth. We can observe the effectiveness of Matching Growth on the large dataset, where the precision decreases slightly and recall increases significantly from 11% to 43%.

Figure 2.14 shows the effectiveness of *FormMatch* on different datasets: small dataset with manual label extraction (*FFC1*), small dataset with automatic label extraction (*FFC1.LX*), large dataset with automatic label extraction (*FFC2.LX*). The F-measure of *FormMatch* decreases for *FFC1.LX* and *FFC2.LX* because of extraction errors. However, the recall of Book in *FFC2.LX* dataset is higher than in *FFC1.LX* dataset. This can be explained as follows. In datasets with a greater number of forms, the similarity signal and correlation signal are better when there is more data. For instance, we can discover additional matches, such as *subject* \sim *category*. However, due to extraction errors, the precision decreases in all three domains. A closer look at the discovered



(a) Without Validation



(b) With Validation

Figure 2.12: The effectiveness of Matching Discovery with different correlation thresholds

matches in *FFC1.LX* and *FFC2.LX* shows that they are similar to the matches discovered in *FFC1*. Since there are errors resulting from the extraction process itself where an element was assigned the wrong label and leads to an incorrect match, we would like to evaluate the result by isolating the extraction error in the following section.

2.4.4.1 Adjusted Evaluation

We compensate for incorrect matches stemming from the label extraction process. In particular, for each incorrect match, we determine whether the error comes from *LabelEx* (by comparing the automatically extracted label and the manually extracted label) or from *FormMatch*. If the error comes from *LabelEx*, we compensate by not counting that error. Figure 2.15 shows the adjusted

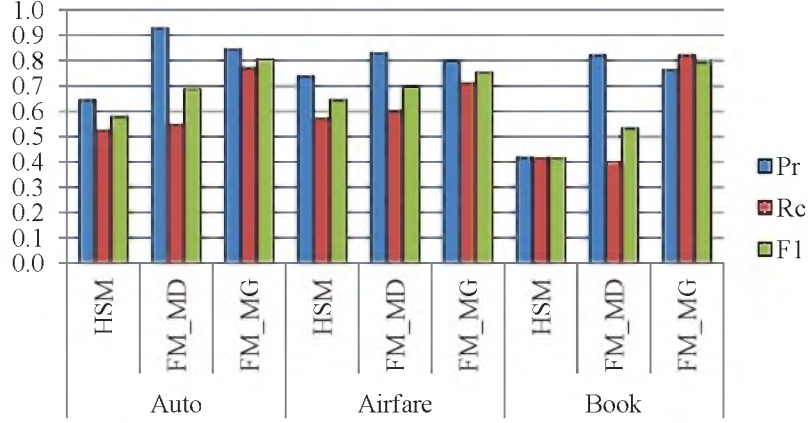


Figure 2.13: Effectiveness of FormMatch on FFC2 dataset with automatic label extraction

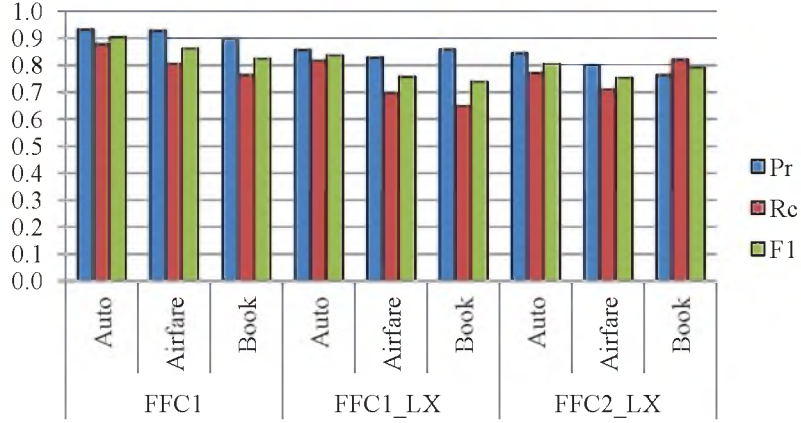


Figure 2.14: Effectiveness of FormMatch on different datasets

measurements. The largest difference is in the Airfare domain (10%) in which LabelEx has the highest error. Comparing the discovered matches to the manually extracted dataset (e.g., *FFC1*), we see that FormMatch is resilient to extraction errors, which is an important requirement for matching large and real datasets given the fact that previous holistic schema-matching approaches were vulnerable and directly affected by the extraction error [53, 93].

2.5 Related Work

There is extensive literature on the topic of database schema-matching [88, 91, 73, 59, 49, 42, 33, 43, 21]. Database schemas often contain useful information for deriving matches. For instance, given two schemas, Melnik et al. [73] leveraged the hierarchy structure and information about data types, primary keys, foreign keys, and so on, to compute the similarity between them. Applying for

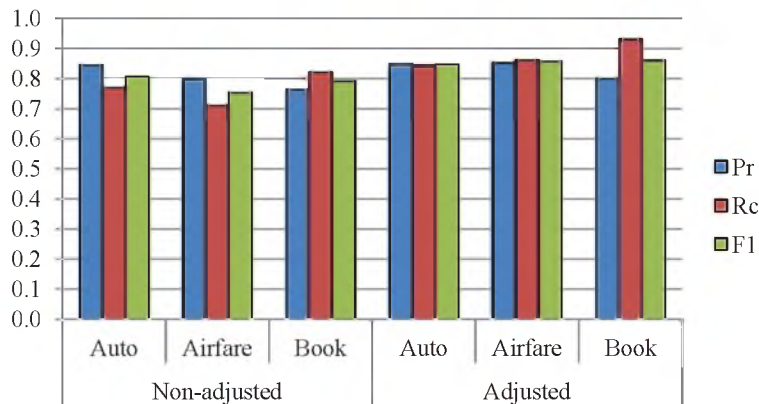


Figure 2.15: Adjusted evaluation

schemas with opaque column names and data values, Kang and Naughton et al. [59] relied on the value distribution of attributes to create a dependency graph for each schema and then used graph matching to find the correspondence between these graphs. Their matching strategy was based on the assumption that all attributes have abundant domain values and attributes in the schemas are strongly-coupled.

Even though Web-form schema-matching is related to the problem of database schema-matching (see e.g., [88, 91]), there are fundamental differences [68]. First, whereas database schemas include useful information about attribute names, data type, value instances, and constraints (key, foreign key), for Web-form schemas, only the association between a label and an element is known. This association, however, is implicit and discovering it can be challenging [77, 108]. Because Web-forms are designed for human consumption, the descriptive labels often vary a lot. In addition, there are often a large number of sites that offer the same services or sell the same products, and there are many form attributes that are not associated with any values. Recognizing these differences, a number of approaches have been proposed for matching forms. These can be broadly classified into three classes: instance-based, clustering-based, and statistical.

Instance-based approaches, as their name implies, rely on the contents hidden behind the form interfaces in order to perform probing and infer a matching [102, 106]. Wang et al. [102] proposed an *instance-based* approach that consists of an ensemble of 3-layers of schemata: a manually-defined global schema, a form-interface schema, and a result schema (extracted by a wrapper). They rely on probing to exhaustively submit all attribute values of the sample records to each input element of the search interface and count the re-appearance of each query value in the resulting pages. This approach is costly, requiring substantial network bandwidth. Besides, a global schema and web instances are expensive to create and subject to constant change. Similarly, WebIQ [106]

retrieves instances from the Web by using hyponymy patterns, e.g., “... such as NP1, NP2” and exploiting the sentence completion feature of search engines. These instances are used to support the schema-matching task. Because of using instances from the Web, a potential problem of WebIQ is that those instances may be biased toward popular instances or noisy data from the Web.

Clustering-based approaches, like [107, 55, 86], use only information visible in each form interface to define a similarity function which combines different element features to measure the distance between two elements. While Wu et al. [107] and He et al. [55] used predefined coefficients, Pei et al. [86] leveraged the distribution of the Domain Cluster and Syntactic Cluster to determine weights of linguistic and domain similarity: the more elements in the domain cluster, the higher the coefficient of linguistic similarity in that cluster. The idea was to use certain attributes in a domain cluster to resolve uncertain attributes in a syntactic cluster. However, the distribution of these clusters varies across different domains and this approach will have less impact when the domain values are scarce or when they are not available. Besides the similarity function, clustering approaches often require the number of clusters or a stopping threshold beforehand. In contrast, PruSM is a data-driven process, and as such, it can naturally reveal the shape of the clusters based on the co-occurrence patterns of attributes and their internal interactions.

To identify *synonyms*, some of these approaches use Wordnet [55] while others leverage only domain values [86]. However, by using WordNet, it is not possible to identify domain-specific synonyms (like `Vehicle` and `Make`), and values are not helpful for attributes that range over a sparse domain, or for attributes that are not associated with any values. Furthermore, these approaches are limited to identifying 1:1 mappings, except for [107], which does derive 1:m mappings. Wu et al. [107] supported complex matching by modeling form interfaces as ordered trees. These trees are first used to identify initial complex mappings. HAC is applied to cluster the remaining attributes to find all 1:1 mappings, which are combined with initial complex mappings to obtain additional complex mappings (using the “bridging effect”). However, constructing an ordered tree for *each* form interface is expensive and the effectiveness of this approach is highly dependent on the quality of this structure, which for the experiments discussed in the paper was manually constructed. In addition, user interactions were required to reconcile uncertain mappings. It is worthy of note that most of these approaches work with (and have only been tested with) a very *small* number of sources, and require noise-free data as input.

Most related to `FormMatch` are the *statistics-based* or *holistic* approaches, which benefit from considering a large number of schemata simultaneously [52, 53, 93], as opposed to pair-wise matching. The term ‘holistic’ was used in the works by Chang, Lochovsky, and Gal [53, 93, 43]. A limitation shared by these approaches is that they require clean data: their performance decreases

significantly when the input data are noisy. As a result, they cannot be directly applied to large, heterogeneous form collections, such as, for example, forms obtained by a Web crawler. Another important distinction between these approaches and *FormMatch* is that they aim to identify attribute synonyms, whereas *PruSM* addresses the more general problem of identifying *all* correspondences among elements. In particular, DCM [53] and HSM [93] derive complex matches by mining positive and negative correlation between form elements. DCM exploits the “a priori” property closure [3] to discover positively and negatively correlated groups. It then selects the highest negatively correlated matches and eliminates matches that are inconsistent with the chosen ones. HSM uses a greedy algorithm based on negative correlation to discover element synonyms. However, DCM and HSM share two critical limitations: they require clean data that limit their scalability and they ignore *rare* attributes, which are inevitably commonplace in large collections of forms. In the presence of rare attributes and noisy data, their performance decreases significantly [53, 93]. Only finding 1:1 mappings and assuming a simple attribute model, MGS [52] assumes a hypothesis that labels are generated by a hidden generative model that contains only a few schema concepts, and each concept is composed of synonym attributes with different probabilities. MGS exhaustively generates all possible models and uses statistical hypothesis tests to select a good one. MGS evaluates and chooses the best global schema model (all at once) while we explore one match at a time. However, given a large number of attributes on a real dataset, considering all their combinations would be expensive.

COMA++ [49] is a schema-matching framework that supports both name-based and instance-based matchers. In contrast to this pair-wised approach, we consider a collection of schemata and attempt to reach a *consensus* terminology for a domain. In addition to schema and instance information, we also exploit statistical information from the attribute co-location patterns. LSD [32] is a machine learning learning-based approach that uses a meta-learner to combine a set of learners. However, LSD requires a mediated schema and human users to manually construct the semantic mappings in the training examples so that the weights can be learned. Given the heterogeneity of a large number of Web-forms where the importance of different features varies a lot in different forms and in different domains, creating the training data is expensive.

To avoid error propagation, *FormMatch* proceeds in two phases: Matching Discovery and then Matching Growth. Although two-phase matching has been used for matching ontology, database schemata and Web-forms [55, 56, 21], they assumed certain matchers are strong and combine them in a fixed manner. However, since the importance of different features can be tied to individual attributes in a form, the contribution of different features often varies for different forms and different domains, and any fixed matcher and its linear combination are unlikely to work well for

all domains. Therefore, techniques that assume that certain matchers are always strong would fail in this scenario. They would also fail for domains where very few attributes have associated values, as in the Book domain where most form elements are text boxes.

2.6 Discussion

2.6.1 Noise and Rare Attributes

We should note that previous approaches to Web-form integration make (strong) assumptions about the input data. Notably, all forms to be integrated belong to the same domain, the label extraction from HTML forms is good, and the labels are normalized. In the experiments reported in the literature, the forms are collected manually; labels are extracted manually. In addition, statistical schema-matching approaches [52, 53, 93] assume a very *small tail* in the label distribution and ignore rare attributes with frequency less than 20%. However, consider Figure 2.3, which shows the long tail label histogram in three different form domains: Auto, Airfare, and Books, i.e., a few attributes have high frequency while many attributes have low frequency. Without preprocessing, the percentage of attributes whose frequency is less than 20% is very high, for example, 83% in Auto domain. Therefore, the small tail Zipf-like distribution can be obtained only by manually preprocessing the data to remove irrelevant terms and simplify the labels. However, as we have illustrated in Example 1, automatically simplifying labels is not trivial and is error prone. Without preprocessing, there would be more rare attributes. Consequently, the applicability of these techniques is greatly reduced [93, 53]. For instance, the matching accuracy can be reduced by as much as 39% and recall can be reduced by as much as 44% in the *absence of manual preprocessing* (Table 2.4) or the average matching accuracy can be reduced by as much as 52% and the average recall can be reduced by as much as 42% when considering more *infrequent attributes*, e.g., attributes that appear as low as 5% of the forms in a collection (Table 2.5).

Furthermore, the integration problem is compounded by errors stemming from previous auto-

Table 2.4: Effectiveness of DCM reduced when not applying preprocessing

<i>Domain</i>	P_T (20%)	R_T (20%)	P_T (10%)	R_T (10%)
Books	0.79 (-0.21)	1	0.74 (-0.26)	1
Airfares	1	1	0.81 (-0.19)	0.82 (+0.11)
Movies	1	1	0.87 (-0.13)	1
MusicRecords	0.93 (-0.07)	1	0.70 (-0.06)	1
Hotels	0.66 (-0.20)	1	0.47 (-0.39)	0.46 (-0.41)
CarRentals	1 (+0.28)	0.63 (-0.37)	1 (+0.28)	0.16 (-0.44)
Jobs	0.70 (-0.30)	1 (+0.14)	0.52 (-0.26)	0.87
Automobiles	1	1	0.66 (-0.27)	0.68 (-0.32)

Table 2.5: Effectiveness of HSM and DCM reduced when considering rare attributes

Domain	$T_c = 20\%$		$T_c = 10\%$		$T_c = 5\%$	
	P_T	R_T	P_T	R_T	P_T	R_T
Airfares	1	1	1	.94	.90	.86
Automobiles	1	1	1	1	.76	.88
Books	1	1	1	1	.67	1
CarRentals	1	1	.89	.91	.64	.78
Hotels	1	1	.72	1	.60	.88
Jobs	1	1	1	1	.70	.72
Movies	1	1	1	1	.72	1
MusicRecords	1	1	.74	1	.62	.88
Average	1	1	.92	.98	.70	.88

(a) HSM with $T_g = 0.9$

Domain	$T_c = 20\%$		$T_c = 10\%$		$T_c = 5\%$	
	P_T	R_T	P_T	R_T	P_T	R_T
Airfares	1	1	1	.71	.56	.51
Automobiles	1	1	.93	1	.67	.78
Books	1	1	1	1	.45	.77
CarRentals	.72	1	.72	.60	.46	.53
Hotels	.86	1	.86	.87	.38	.34
Jobs	1	.86	.78	.87	.36	.46
Movies	1	1	1	1	.48	.65
MusicRecords	1	1	.76	1	.48	.56
Average	.95	.98	.88	.88	.48	.58

(b) DCM

matic processes used to construct the input. For instance, as any automatic interface extraction cannot be perfect, it will likely introduce some noise (i.e., erroneous extraction), which challenges the performance of the subsequent matching algorithm. As reported in [53], the matching quality e.g., the accuracy of the base DCM framework, degraded to 30% with extraction errors amounting to only 15%. Thus, an important requirement is that the approach must address the robustness problem in integrating the interface extraction step and the matching algorithm.

To deal with noise, DCM proposed to run multiple trials on different samples and ensemble the results. In contrast, using aggregation and the prudent strategy makes `FormMatch` less vulnerable to the extraction errors, making it more robust and resilient to incorrect extractions. Additionally, Matching Growth incrementally finds additional matches for infrequent attributes, thereby improving recall without significant reduction in accuracy. With the prudent approach, among important achievements of `FormMatch` are its robustness against inherent noise from previous steps in the integration process and its scalability due to the fact that it does not require manual preprocessing. Our experiment on a real dataset with labels that were automatically extracted by an extraction program shows that `FormMatch` can derive high quality matches.

2.6.2 Limitation

`FormMatch` cannot find a match if either its label similarity or value similarity is not sufficient enough. Relaxing the constraints and buffering the candidates to process later can help to improve recall but less guarantee for precision. In contrast, the constraints may become complicated if we consider more features. A learning-based [32] or self-learning-based approach [76] is an available option to combine the constraints.

2.6.3 Future Work

One potential source of improvement for `FormMatch` includes combining more information like element proximity, DOM structure, element name, and type to find the most confident match. By bootstrapping `FormMatch` with a better starting set of labels, its effectiveness can be improved. We would like to experiment with alternative procedures to automatically simplify the labels and perform syntactic merging, and study their trade-offs. To improve the quality of derived matches, we would like to use a “look-ahead” buffer which considers the top-k highest attribute pairs to augment the correlation scores and improve imperfect orderings.

CHAPTER 3

MATCHING MULTILINGUAL SCHEMATA IN WIKIPEDIA

3.1 Introduction

With over 17.9 million articles and 10 million page views per month [104], Wikipedia has become a popular and important source of information. One of its most remarkable aspects is *multilingualism*: there are Wikipedia articles in over 270 languages. This opens up new opportunities for knowledge sharing among people that speak different languages both within and outside the scope of Wikipedia. For example, cross-language links, that connect an article in one language to the corresponding article in another, have been used to derive better translations in cross-language information retrieval and machine translation [39, 75, 87, 89]. Even though many languages are represented in Wikipedia, the geographical distribution of Wikipedia users is highly skewed. One of the explanations for this effect is that many languages, including languages spoken by large segments of the world population, are underrepresented. For example, there are 328 million English speakers worldwide and 20% of the Wikipedia pages are in English; in contrast, there are 178 million Portuguese speakers and only 3.75% of Wikipedia articles are in Portuguese. Recognizing this problem, there are a number of ongoing efforts that aim to improve access to Wikipedia content. By leveraging the existing multilingual Wikipedia corpus, techniques have been proposed to: combine content provided in documents from different languages and thereby improve their documents [2, 20]; find missing cross-language links [85, 92]; aid in the creation of multilingual content [64]; and help users who use different languages to search for named entities in the English Wikipedia [97].


Besides textual content, Wikipedia has also become a prominent source for structured information. A growing number of articles contain an *infobox* that provides a structured record e.g., a set of attribute value pairs summarizing important information for the entity described in the article. This information has been used to support richer queries over Wikipedia content [7, 62, 78, 19]. While much work has been devoted to supporting structured queries, no previous effort has looked into providing support for *multilingual* structured queries. Here, we examine the problem of *matching schemas of infoboxes represented in different languages*, a necessary step for supporting these

queries.

By discovering multilingual attribute correspondences, it is possible to integrate information from different languages and to provide more complete answers to user queries. A common scenario is when the answer to a query cannot be found in a given language but it is available in another. In a study of the 50 topics used in the GikiCLEF [44], just 9 topics had answers in all 10 languages used in the task [25]. However, almost every query had an answer in the English Wikipedia. Thus, by supporting multilingual queries and providing the relevant English documents as part of the answer, recall could improve for most other languages. In addition, some queries can benefit from integrating information from infoboxes represented in different languages. Consider the query: *Find the genre of and the studio that produced the film “The Last Emperor”*. To provide a complete answer to this query, we need information from the two infoboxes in Figure 3.1.

There are several challenges involved in finding multilingual correspondences. Even within a language, finding attribute correspondences is difficult. Although authors are encouraged to provide some structure in Wikipedia articles, e.g., by selecting appropriate templates and categories, they often do not follow the guidelines or follow them loosely. This leads to several problems, in particular, schema heterogeneity—the structure of infoboxes for the same entity type (e.g., actor, country) is different for different instances. Both polysemy and synonymy are observed among attribute names. A given name can have different semantics (e.g., *born* can mean *birth date* or *country of birth*) and different names can have the same meaning (e.g., *alias* and *other names*). This problem is compounded when we consider multiple languages. Figure 3.1 shows an example of heterogeneity in infoboxes describing the same entity in different languages. Some attributes in the English infobox do not have a counterpart in the Portuguese infobox and vice-versa. For instance: *produced by*, *editing by*, *distributed by*, and *budget* are omitted in the Portuguese version, while *gênero* (genre) is omitted in the English version. An analysis of the overlap among attribute sets from infoboxes in English and Portuguese (see Table 3.1) shows that on average, only 42% of the attributes are present in both languages. Besides the variation in structure, there are also inconsistencies in the attribute values; for example, *running time* is 160 minutes in the English version and 165 minutes in the Portuguese version, where it is not presented as a separate (and named) attribute; Ryuichi Sakamoto appears under *Music by* in English and under *Elenco original* (cast) in Portuguese.

To identify multilingual matches, a possible strategy would be to translate the attribute names (and values) using a multilingual dictionary or a machine translation system and then apply traditional schema or ontology matching techniques [88, 36, 40]. However, this strategy is limited since in many cases, the correct correspondence is not found among the translations. For example, in

Directed by	Bernardo Bertolucci	The Last Emperor	
Produced by	Jeremy Thomas		
Written by	Mark Peploe Bernardo Bertolucci	O Último Imperador (PT/BR)	
Starring	John Lone Joan Chen Peter O'Toole	 Reino Unido / Itália França / China 1987 • cor • 165 min	
Music by	Ryuichi Sakamoto	Produção	
Cinematography	Vittorio Storaro	Direção	Bernardo Bertolucci
Editing by	Gabriella Cristiani	Roteiro	Mark Peploe / Bernardo Bertolucci
Studio	Hemdale Film	Elenco original	John Lone Joan Chen Peter O'Toole Ryuichi Sakamoto
Distributed by	Columbia Pictures	Gênero	drama biográfico / épico
Release date(s)	United States: November 18, 1987	Idioma original	inglês / mandarim / japonês
Running time	160 minutes (theatrical) 218 minutes (television)	IMDb: (inglês)  (português) 	
Country	China Italy United Kingdom France	Projeto Cinema • Portal Cinema	
Language	English Mandarin Chinese		
Budget	\$23.8 million ^[1]		

(a) English

(b) Portuguese

Figure 3.1: Excerpts from English and Portuguese infoboxes for the Film *The Last Emperor*.**Table 3.1:** Overlap in infoboxes

	film	show	actor	artist	channel	company	comics ch.	album	adult actor	book	episode	writer	comics	fictional ch.
Pt-En	36%	45%	42%	52%	15%	31%	59%	52%	47%	38%	31%	63%	47%	32%
Vn-En	87%	75%	46%	67%										

articles describing movies, the correct alignment for the English attribute *starring* is the Portuguese attribute *elenco original*. However, the dictionary translation is *estrelando* for the former and *original cast* for the latter, and neither is used in the Wikipedia infobox templates to name an attribute. WordNet is another source of synonyms that can potentially help in matching, but its versions in many languages are incomplete. For instance, the Vietnamese WordNet covers only 10% of the senses present in the English WordNet. Furthermore, traditional techniques such as string similarity may fail even for languages that share words with similar roots. Consider the term

editora, which in Portuguese means *publisher*. Using string similarity, it would be very close to *editor*, but this would be a false cognate.

Recently, techniques have been proposed to identify multilingual attribute alignments for Wikipedia infoboxes. However, these have important shortcomings in that they are designed for languages that share similar words [2, 20], or demand a considerable amount of training data [2]. Consequently, they cannot be effectively applied to languages with distinct representations or different roots; and their applicability is also limited for underrepresented languages in Wikipedia, which have few pages and thus, insufficient training data.

We propose WikiMatch, a new approach to multilingual schema-matching that addresses these limitations. WikiMatch leverages *latent semantic analysis* [67] and information available in Wikipedia. These different sources of similarity information are combined in a systematic manner so as to prioritize the derivation of high-confidence attribute alignments. These alignments are then used to help find additional matches. Our alignment algorithm identifies, in a single step, both intralanguage and interlanguage correspondences, and it also identifies one-to-many correspondences. Because WikiMatch does not require training data, it is able to handle underrepresented languages; and since it does not rely on string similarity on attribute names, it can be applied both to similar and morphologically distinct languages. Furthermore, it does not require external resources such as bilingual dictionaries, thesauri, ontologies, or automatic translators.

3.2 Problem Definition and Solution Overview

3.2.1 Problem Definition

A Wikipedia article is associated with and describes an *entity* (or object). Let A be an article in language L associated with entity E . Among the different components of A , here, we are interested in its *title*; *infobox*, which consists of a structured record that summarizes important information about E ; and *cross-language links*, URLs of pages in languages other than L that describe E . An infobox I contains a set of attribute-value pairs $\{\langle a_1, v_1 \rangle, \dots, \langle a_n, v_n \rangle\}$. Figure 3.1(a) shows the infobox of an English article with 14 attribute-value pairs. Since there is a one-to-one relationship between I and its associated E , we use these terms interchangeably in the remainder of the chapter. We define the set of attributes in an infobox I as the *schema of I* (S_I).

The value v of an attribute a in an infobox I may contain one or more hyperlinks to other Wikipedia entities. For example, in Figure 3.1(a), the value for the attribute *Directed by* contains a hyperlink to the entity *Bernardo Bertolucci*. We denote such a hyperlink by the tuple $h = (I, v, J)$, where J is the infobox pointed to by v . We distinguish between hyperlinks that point to another entity in the same language (which define *relationships*) and hyperlinks that point to articles describing the same entity in different languages. We refer to the latter as *cross-language links*: $cl = (I_L, I_{L'})$.

These links can be found in most articles and are located on the pane to the left of the article. For example, there are cross-language links between the English and Portuguese articles in Figure 3.1.

An article is also associated with an *entity type* T that identifies the *type* to which an article belongs. For example, the article in Figure 3.1(a) belongs to the entity type “Film”. There are different ways to determine the entity type for an article, including from the categories defined for the article [95]; from the template defined for the infobox [17]; or from the structure of the infobox [82]. Given a set I_L of infoboxes in language L associated with entity type T , we refer to the set of all distinct attributes in I_L as the *schema of* T (S_T). Given two infoboxes I_L and $I_{L'}$ with type T that are connected by a cross-language link, we refer to the union of the attributes in their schemas, $S_D = S_I \cup S_{I'}$, as a *dual-language infobox schema*.

The problem we address can be defined as follows: Given two sets of infoboxes I_L and $I_{L'}$ in languages L and L' , respectively, such that both sets are associated with the entity type T , and whose infoboxes are connected through cross-language links, to match S_T and $S_{T'}$, the schemas of infoboxes in the two sets, we need to find correspondences (or matches) $\langle a, a' \rangle$ such that a is an attribute of S_I , a' is an attribute of $S_{I'}$, and a and a' have similar meaning.

3.2.2 Solution Overview

The WikiMatch framework is shown in Figure 3.2. After aggregating attributes having the same name from the infobox repository, for every two attributes, we compute cross-language similarity (Section 3.3.1), including cross-language value similarity, cross-language link similarity, and attribute correlation, to derive correspondences (Section 3.3.2) and revise uncertain matches (Section 3.3.2).

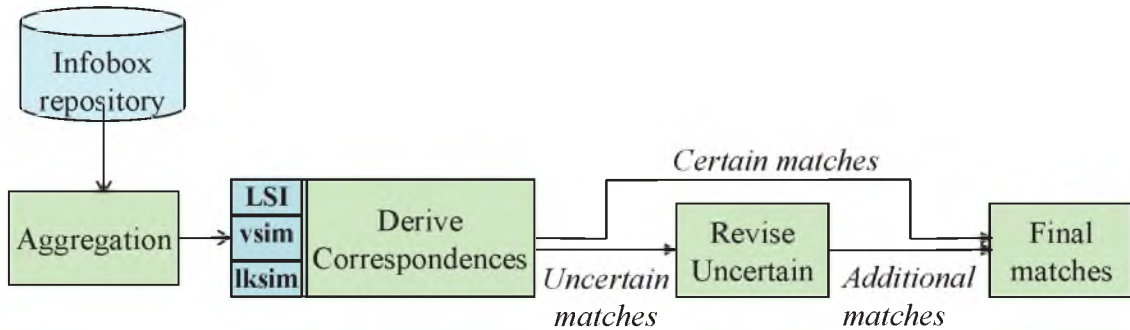


Figure 3.2: Matching Multilingual Infoboxes

3.3 WikiMatch

3.3.1 Computing Cross-Language Similarities

3.3.1.1 Cross-Language Value Similarity

Because of the structural heterogeneity among infoboxes in different languages, by combining their attributes in a unified schema for each distinct type, we gather more evidence that helps in the derivation of correspondences. We also collect for each attribute a in an entity schema S_T , the set of values v associated with a in all infoboxes with type T . Value similarity for two attributes is then computed as the cosine similarity between their value vectors.

Since a concept can have different representations across languages, direct comparison between vectors often leads to low similarity scores. Thus, we use an automatically created translation dictionary to help improve the accuracy of the similarity score: whenever possible, the values are translated into the same language before their similarity is computed. We exploit the cross-language links among articles in different languages to create a dictionary for their titles [85]. The translation dictionary from a language L to language L' is built as follows. For each article A in L with a cross-language link to article A' in L' , we add an entry to the dictionary that translates the title of article A to the title of article A' .

Given an attribute a with value vector v_a in language L , an attribute a' with value vector $v_{a'}$ in language L' , and a translation dictionary D , we construct the translated value vector of a as follows: if a value of v_a can be found in D , we replace it by its representation in L' . We denote the translated value vector of a as v_a^t , and define the *value similarity* between a and a' as:

$$vsim(a, a') = \cos(v_a^t, v_{a'}) \quad (3.1)$$

where the vector components are the raw frequencies (tf).

Example 8. Given the vectors for *nascimento* and *born* respectively as: $v_a = \{1963, \text{Irlanda}:1, 18 \text{ de Dezembro } 1950:1, \text{Estados Unidos}:1\}$ and $v_{a'} = \{1963, \text{Ireland}:1, \text{June } 4 \text{ } 1975:1, \text{United States}:2\}$, where the numbers after the colons indicate the frequency of each value. Translating v_a , we get $v_a^t = \{1963, \text{Ireland}:1, \text{December } 18 \text{ } 1950:1, \text{United States}:1\}$. Thus, $vsim(v_a, v_{a'}) = \cos(v_a^t, v_{a'}) = 0.71$ ■

3.3.1.2 Link Structure Similarity

The attribute values in the infoboxes often link to other articles in Wikipedia. For example, attribute *Directed by* in Figure 3.1(a) has the value *Bernardo Bertolucci* that links to an article for this director in English. Similarly, the value of attribute *Direto* in Figure 3.1(b) links to an article

for this director in Portuguese. Because of the multilingual nature of Wikipedia, the two articles for *Bernardo Bertolucci* are linked by a cross-language link. Similar to Bouma et al. [20], we leverage this feature as another source of similarity. In this example, the link structure information helps us determine that $\langle Directed\ by, Direo \rangle$ match. We define the *link structure set* of an attribute in an entity type schema S as the set of outgoing links for all of its values. Given two attributes, the larger the intersection between their link structures, the more likely they are to form a correspondence. Two values are considered equal if their corresponding landing articles are linked by a cross-language link. Let $ls(a) = \{l_a^i | i = 1..n\}$ and $ls(a') = \{l_{a'}^j | j = 1..m\}$ be the link structure sets for attributes a and a' . The *link structure similarity* $lsim$ between these attributes is measured as follows:

$$lsim(a, a') = \cos(ls(a), ls(a')) \quad (3.2)$$

3.3.1.3 Attribute Correlation

The notion of correlation has been successfully applied in strategies to identify correspondences in Web-form schema-matching [53, 81, 93]. There, the intuition was that synonyms should not co-occur in a given form and therefore, they should be negatively correlated. For a given language, the same intuition holds for attributes in an infobox—synonyms should not appear together. However, for identifying cross-language correspondences, the opposite is true: if we combine the attribute names for corresponding infoboxes across languages creating a dual-language infobox schema, cross-language synonyms are likely to co-occur.

Unlike previous work, which applied absolute correlation measures for all attribute pairs, we use Latent Semantic Indexing (LSI) [31] and we do so both for attributes within the same language and across languages. Our inspiration comes from the Cross-language Information Retrieval literature where LSI was one of the first methods applied to match terms across languages [67]. While LSI has traditionally been applied to terms in free text, here we use it to estimate the structural correlation between schema attribute names. LSI is applied to a matrix of terms by documents (in our scenario, a matrix of attribute names by infobox schemas) that contains the occurrences for attributes in each dual-language infobox schema.

Given an entity type T , let $D = \{d_i | i = 1..m\}$ be the set of dual-language infoboxes and $A = \{a_j | j = 1..n\}$, the set of unique attributes in D . Let $M(n \times m)$ be a matrix with n rows and m columns where $M(i, j) = 1$ if attribute a_i appears in dual-language infobox d_j , otherwise $M(i, j) = 0$. Each row in the matrix corresponds to the occurrence pattern of that attribute over D . Figure 3.3 has an example of such a matrix. We apply the truncated singular value decomposition (SVD) [67] to derive $\tilde{M} = U_f S_f V_f^T$ by choosing the f most important dimensions, and vectors \vec{a}_p, \vec{a}_q are scaled by the top f singular values in matrix S .

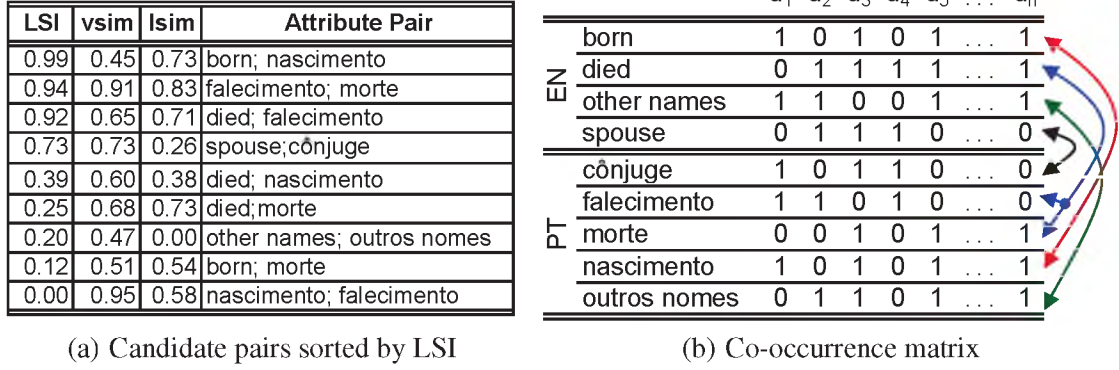


Figure 3.3: Some attributes for *Actor* in Pt-En

After the dimensionality reduction, attributes are no longer independent. If attribute names are used in similar infoboxes, they will have similar vectors in the reduced representation. SVD causes cross-language synonyms to be represented by similar vectors (since they would have many co-occurrences). This is what makes LSI suitable for cross-language matching. To measure the correlation between attributes in different languages, we compute the cosine between their vectors. In contrast, for attributes within the same language, we take the complement of the cosine between their vectors. For attributes in the same language which co-occur in an infobox, we set the LSI score to 0 as they are unlikely to be synonyms. Thus, in WikiMatch, the LSI score for attributes a_p and a_q is computed as:

$$LSI(a_p, a_q) = \begin{cases} \cosine(\vec{a}_p, \vec{a}_q) & \text{if } a_p \text{ in } L \wedge a_q \text{ in } L' \\ 0 & \text{if } a_p, a_q \text{ in } L_L \text{ or } L_{L'} \\ 1 - \cosine(\vec{a}_p, \vec{a}_q) & \text{if } a_p \wedge a_q \text{ in } L \text{ or } L' \end{cases} \quad (3.3)$$

For attributes in the same language, a LSI score of 1 means they never co-occur in a dual-language infobox. Consequently, they are likely to be intralanguage synonyms. In contrast, for attributes in different languages, a LSI score of 1 means they co-occur in every dual-language infobox. Thus, they have a good chance of being cross-language synonyms.

Note that, as illustrated in Figure 3.1, corresponding infoboxes are not parallel, i.e., there is not a one-to-one mapping between attributes in the two languages. As a consequence, LSI is expected to yield uncertain results for cross-language synonyms and when rare attributes are present, the same outcome will be observed for intralanguage synonyms. As we discuss in the experiment, when used in isolation, LSI is not a reliable method for cross-language attribute alignment. However, if combined with the other sources of similarity, it contributes to high recall and precision in the derivation of correspondences.

Important advantages of using LSI for finding cross-language synonyms are: (i) all attribute names are transformed into a language-independent representation, thus there is no need for trans-

lation; (ii) external resources, e.g., dictionaries, thesauri, or automatic translators are not required; (iii) languages need not share similar words; (iv) LSI can implicitly capture the second order of term co-occurrence.

We examined other alternatives for computing attribute correlations, including the measures used in [53, 81, 93]. However, since they were defined to identify synonyms within one language, they cannot be directly applied to our problem. We have extended them to consider co-occurrence frequency in the dual infoboxes, but LSI outperforms all of them. This can be explained in part by the dimensionality reduction achieved by SVD and the consideration of the co-occurrence patterns of LSI for attribute pairs over all dual-language infoboxes.

3.3.2 Deriving Correspondences

The effectiveness of any given similarity measure varies for different attributes and entity types. For example, two attributes may have different values and yet be synonyms, or vice versa. Thus, to derive correspondences, an important challenge is how to combine the similarity measures. Our `AttributeAlignment` algorithm (Algorithm 3) combines different similarity measures in such a way that they *reinforce* each other. It prioritizes the derivation of *confident matches*, i.e., correspondences that are more likely to be correct. Given as input the set of all attributes for infoboxes that belong to a given type, it groups together attributes that have the same label, and for these, combines their values—we refer to the set of such groups as *AG*. The attribute groups in *AG* are then paired with each other, and for each pair, the similarity measures are computed (Section 3.3.1). This step creates a set of tuples that associate similarity values with each attribute pair: $(\langle a_p, a_q \rangle, v_{sim}, l_{sim}, LSI)$. The tuples with a LSI score greater than a threshold T_{LSI} are then added to a priority queue *P*. Intuitively, a pair of matching attributes should have a high positive correlation. However, because of the heterogeneity in the data, this correlation may be weak, so T_{LSI} could be low.

Note that the relative ordering of LSI score is important. Thus, the tuples in *P* are sorted in *decreasing order* of LSI score to ensure the most certain matches are processed first to avoid the early selection of incorrect matches, which can lead to error propagation in future matches. The similarities for a pair of attributes a_p, a_q are combined according to the constraint: if the maximum value of $v_{sim}(a_p, a_q)$ and $l_{sim}(a_p, a_q)$ is greater than a threshold T_{sim} , then $\langle a_p, a_q \rangle$ is a certain candidate correspondence. The intuition is two attributes form a certain correspondence, if they are correlated and this is corroborated by at least one of the other similarity measures. The rationale for defining T_{sim} and T_{LSI} is that the thresholds for the first should be set high to enable the selection of certain correspondences, whereas the second should be set low since it is used to sort the candidate correspondences in the order they will be processed. We have also studied the behavior of `WikiMatch` using different thresholds. As shown later in the experiment, our approach

remains effective and obtains high F-measure values for a broad range of threshold values.

Figure 3.3(a) shows some of the attribute pairs in P , with their similarity scores. $\langle born, nascimento \rangle$ is a certain match because all similarity scores are high. Figure 3.3(b) shows the (reduced) set of attributes in English and Portuguese for the type *Actor*. The cells in this matrix contain the number of occurrences for an attribute in each dual-language infobox. The matches in the ground truth are indicated by the arrows. Notice that *died* matches two attributes in Portuguese.

If a candidate correspondence $\langle a_p, a_q \rangle$ does not satisfy the constraint in line 10 of Algorithm 3, it is added to the set of uncertain matches U (line 13) to be considered later (Section 3.3.3). Otherwise, if it does satisfy the constraint, it is given as input to `IntegrateMatches` (Algorithm 4), which decides whether it will be integrated into an existing match, originate a new one, or be ignored. `IntegrateMatches` outputs a set of matches, M , where each match $m = \{a_1 \sim \dots \sim a_m\}$ includes a set of synonyms, both within and across languages. `IntegrateMatches` takes advantage of the correlations among attributes to determine how to integrate the new correspondence into the set of existing matches.¹ If neither of the attributes in the new correspondence appears in the existing matches M , a new matching component is created (line 5). If at least one of the attributes is already in a match m_j in M , for example, suppose a_p appears in m_j , and the LSI score between a_q and all attributes a_j in m_j is greater than the correlation threshold T_{LSI} (line 8), then a_q becomes a new element in m_j (line 9, where $+ \sim \{a_q\}$ denotes that a_q is added to the existing match m_j), otherwise, it is ignored. Since T_{LSI} is set low, the requirement of having positive correlations with all attributes in an existing match is not too strict and helps merging intra- and interlanguage synonyms. Relaxing this constraint to some and not all attributes in the match could lead to incorrect matches. Since our correlation measures work for attribute pairs both within and across languages, `IntegrateMatches` can discover both intra- and cross-language synonyms. This is shown in the example below.

Example 9. Consider the attribute pairs in Figure 3.3(a) for type *Actor*, ordered by descending LSI scores, with $T_{LSI}=0.1$. Assume that the set of existing matches M includes $m = \{\text{died} \sim \text{falecimento}\}$, and we have two candidate pairs, $p_1 = \langle \text{died}, \text{morte} \rangle$ and $p_2 = \langle \text{died}, \text{nascimento} \rangle$. Since the LSI score for “*morte*” and “*falecimento*” $\geq T_{LSI}$, “*morte*” is integrated into m , i.e., $m = \{\text{died} \sim \text{falecimento} \sim \text{morte}\}$. In contrast, p_2 is not added to m since the LSI score for “*falecimento*” and “*nascimento*” is zero as they are in the same language and often co-occur. ■

¹This algorithm is similar to the Algorithm 2 in Chapter 2, but it was simplified to not consider grouping attributes.

Algorithm 3 AttributeAlignment

```

1: Input: Set of infobox attributes for an entity type  $T$ 
2: Output: Set of matches  $M$ 
3: begin
4:  $M \leftarrow \emptyset, P \leftarrow \emptyset$ 
5: for each pair  $\langle a_p, a_q \rangle$  such that  $a_p, a_q \in AG$  do
6:   Compute  $vsim, lsim, LSI$ 
7:    $P \leftarrow P \cup (\langle a_p, a_q \rangle, vsim, lsim, LSI) \mid LSI > T_{LSI}$ 
8: while  $P \neq \emptyset$  do
9:   Choose pair  $\langle a_p, a_q \rangle$  with the highest  $LSI$  score from  $P$ 
10:  if  $\max(vsim(a_p, a_q), lsim(a_p, a_q)) > T_{sim}$  then
11:     $M \leftarrow \text{IntegrateMatches}(\langle a_p, a_q \rangle, M)$ 
12:  else
13:     $U \leftarrow \langle a_p, a_q \rangle$  /*buffering uncertain matches*/
14:    Remove  $\langle a_p, a_q \rangle$  from  $P$ 
15:   $U' \leftarrow \text{ReviseUncertain}(U)$ 
16:  for each  $u \in U'$  do
17:     $M \leftarrow \text{IntegrateMatches}(u, M)$ 
18: end

```

Algorithm 4 IntegrateMatches

```

1: Input: candidate pair  $\langle a_p, a_q \rangle$ , set of current matches  $M$ 
2: Output: updated set of matches  $M$ 
3: begin
4: if neither  $a_p$  nor  $a_q \in M$  then
5:    $M \leftarrow M + \{a_p \sim a_q\}$ 
6: else if either  $a_p$  or  $a_q$  appears in  $M$ 
7:   /* suppose  $a_p$  appears in  $m_j$  and  $a_q$  does not appear */
8:   for each  $a_j \in m_j$ , s.t.  $LSI_{qj} > T_{LSI}$  do
9:      $m_j \leftarrow m_j + (\sim \{a_q\})$ 
10: end

```

3.3.3 Revising Uncertain Matches

Because our matching strategy prioritizes correspondences for which it has high confidence, it may leave out matches that are correct but that have low confidence—the uncertain matches. Consider, for example, value similarity. Although *born* and *morte* (*died*) are not equivalent, their similarity is noticeable since they share many values and links as both attributes contain dates and places. On the other hand, although *outros nomes* and *other names* are equivalent, their value similarity is low as they do not share values or links. Thus, although high value similarity provides useful evidence for deriving attribute correspondences, it may also prevent correct matches from being identified. The `ReviseUncertain` step uses the set M of matches derived by `AttributeAlignment` (line 15) to help identify additional matches, by reinforcing or negating the uncertain candidates (in set U). A challenge in this step is how to balance the potential gain in recall with a potential loss in precision. Our solution to this problem is to consider only the subset U' of attribute pairs in U whose attributes are highly correlated with the existing matches. To capture this, we introduce the

notion of *inductive grouping score*. Let $\langle a, a' \rangle$ be an uncertain correspondence in U , and let C_a and $C_{a'}$ be the set of matched attributes co-occurring with a and a' , respectively, in their monolingual schemas. The inductive grouping score between a and a' is the average grouping score of a and a' with each attribute in C_a and $C_{a'}$:

$$\tilde{g}(a, a') = \frac{1}{|C|} \sum_{c_a \in C_a, c'_a \in C_{a'} | c_a \sim c'_a} g(a, c_a) * g(a', c'_a) \quad (3.4)$$

where the grouping score g is computed as follows:

$$g(a_p, a_q) = \frac{O_{pq}}{\min(O_p, O_q)} \quad (3.5)$$

O_p and O_q are the number of occurrences of attributes a_p and a_q , and O_{pq} is the number of times they co-occur in the set of infoboxes. Note that the grouping score is computed for the schemas of the two languages separately. The inductive grouping score is high if a_p and a_q co-occur often with the attributes in the discovered matches. The final step is to integrate revised matches (lines 16-18). We take advantage of the certain matches in M to validate the revised matches U' : `IntegrateMatches` is invoked again, but this time it considers pairs with similarity lower than T_{sim} . Although we could first threshold on different values of T_{sim} , as we discuss in Section 3.4.3, revising uncertain matches as a separate step helps improving recall while maintaining high precision for a wide range of T_{sim} .

Example 10. Consider the attribute pairs in Figure 3.3(a), let $M = \{\text{born} \sim \text{nascimento}, \text{spouse} \sim \text{cnjuge}\}$ be the set of existing matches. The pairs $\langle \text{other names}, \text{outros nomes} \rangle$ and $\langle \text{born}, \text{morte} \rangle$ are uncertain candidates since their value similarities are lower than the threshold. If the attributes in these pairs co-occur often with *born* and *spouse*, the inductive grouping scores \tilde{g} of $\langle \text{other names}, \text{outros nomes} \rangle$ and $\langle \text{born}, \text{morte} \rangle$ are high, and thus, these candidate matches will be revised and added to U' . Since $\{\text{born} \sim \text{nascimento}\}$ has been identified as a match, *morte* cannot be integrated into this match because *morte* and *nascimento* are in the same language and co-occur in infoboxes (so their LSI score is zero). In contrast, neither *outros nomes* nor *other names* appear in M , so this pair can be added as a new match. ■

3.4 Experimental Evaluation

3.4.1 Dataset and Evaluation Metrics

3.4.1.1 Dataset

We collected Wikipedia infoboxes related to movies from three languages: English, Portuguese, and Vietnamese. Our aim in selecting these languages was to get variety in terms of morphology and in the number of infoboxes. Portuguese and English share words with similar roots, while Vietnamese is very different from the other two languages; and there are significantly fewer infoboxes

for the pair Vietnamese-English (Vn-En) than for Portuguese-English (Pt-En). This is also reflected in the number of types covered by the Vietnamese infoboxes (see below). We selected Portuguese and Vietnamese infoboxes that belong to articles which have cross-language links to the equivalent English article. The dataset for the Pt-En language pair consists of 8,898 infoboxes, while there are 659 infoboxes for the Vn-En pair. For each language, we group infoboxes that belong to the same entity type by using WIClust (see Chapter 5). There are 14 such groups for Pt-En, and 4 for Vn-En.

We created the ground truth for all entity types in the dataset. A bilingual expert labeled as correct or incorrect all the correspondences containing attributes from two distinct languages. A pair of attributes $\langle a, a' \rangle$ is considered a correct alignment if a and a' have similar meaning. The ground truth set for the Pt-En pair has 315 alignments while the Vn-En pair has 160 alignments.

3.4.1.2 Evaluation Metrics

We use weighted scores to account for the importance of different attributes and, consequently, of the matches involving them. Intuitively, a match between frequent attributes will have a higher weight. Let C be the set of cross-language matches derived by our algorithm; \mathcal{G} be the cross-language matches in the ground truth; S_T be the set of attributes of entity type T in language L ; and S'_T be the attributes in language L' of the corresponding type of T . Given an attribute $a_i \in S_T$, we denote by $c(a_i)$ and $c_G(a_i)$ the set of attributes in S'_T that correspond to a_i in C and \mathcal{G} , respectively. Let A_C and $A_{\mathcal{G}}$ be the set of attributes in S_T that appear in C and \mathcal{G} , respectively. The *weighted scores* are computed as follows:

$$Precision = \sum_{a_i \in A_C} \frac{|a_i|}{\sum_{a_k \in A_C} |a_k|} Pr(c(a_i)) \quad (3.6)$$

$$Recall = \sum_{a_i \in A_{\mathcal{G}}} \frac{|a_i|}{\sum_{a_k \in A_{\mathcal{G}}} |a_k|} Rc(c(a_i)) \quad (3.7)$$

$$Pr(c(a_i)) = \sum_{a'_j \in c(a_i)} \frac{|a'_j|}{\sum_{a'_k \in c(a_i)} |a'_k|} * correct(a_i, a'_j) \quad (3.8)$$

$$Rc(c(a_i)) = \sum_{a'_j \in c_G(a_i)} \frac{|a'_j|}{\sum_{a'_k \in c_G(a_i)} |a'_k|} * correct(a_i, a'_j) \quad (3.9)$$

where $|a_i|$ represents the frequency of attribute a_i in the infobox set; $correct(a_i, a'_j)$ returns 1 if the extracted correspondence $\langle a_i, a'_j \rangle$ appears in \mathcal{G} and 0 otherwise. Similar to Chang and He et al. [53], we compute precision and recall as the weighted average over the precision and recall of each attribute a_i (Equation 3.6 and 3.7), and the precision and recall of attribute a_i are also averaged by the contribution of each attribute a'_j in S'_T which corresponds to a_i (Equation 3.8 and 3.9). We compute F-measure as the harmonic mean of precision and recall. We illustrate these measures in Example 11.

Example 11. Consider $S_T = \{a_1, a_2\}$, $S'_T = \{a'_1, a'_2, a'_3\}$, and associated frequencies (0.6, 0.4) and (0.5, 0.3, 0.2). Suppose $G = \{\{a_1 \sim a'_1 \sim a'_2\}, \{a_2 \sim a'_3\}\}$, and the alignment algorithm derives $M = \{\{a_1 \sim a'_1\}, \{a_2 \sim a'_3\}\}$. We have $c(a_1) = \{a'_1\}$, $c(a_2) = \{a'_3\}$, while $c_G(a_1) = \{a'_1, a'_2\}$, $c_G(a_2) = \{a'_3\}$. Therefore:

$$pr(c(a_1)) = \frac{0.5}{0.5} * correct(a_1, a'_1) = 1 \text{ and } pr(c(a_2)) = 1;$$

$$Precision = \frac{0.6}{0.6+0.4} * pr(c_1) + \frac{0.4}{0.4+0.6} * pr(c_2) = 1;$$

$$rc(c(a_1)) = \frac{0.5}{0.5+0.3} * correct(a_1, a'_1) + \frac{0.3}{0.5+0.3} * correct(a_1, a'_2) \\ = \frac{0.5}{0.8} * 1 + \frac{0.3}{0.8} * 0 = 0.625, \text{ and } rc(c_2) = 1;$$

$$Recall = \frac{0.6}{0.6+0.4} * rc(c(a_1)) + \frac{0.6}{0.6+0.4} * rc(c(a_2)) = 0.775. \quad \blacksquare$$

3.4.2 Comparison against Existing Approaches

We compared WikiMatch to techniques from schema-matching, cross-language information retrieval, and to a system designed to align and complete Wikipedia Templates across languages. They are described below.

–*LSI*. We use *LSI* [31] as a technique for cross-language attribute alignment. LSI similarity scores were computed for all attribute pairs $\{a_p, a_q\}$ in an entity type T , where $a_p \in L$ and $a_q \in L'$. The top 1, 3, 5, and 10 scoring correspondences for each a_p were used to identify matches. The best F-measure value was obtained by the top-1 configuration.

–*Bouma*. This approach for aligning infobox attributes across languages uses attribute values and cross-language links [20]. The input to Bouma was the same provided to WikiMatch, i.e., attributes grouped by their entity types.

–*COMA++* [9]. This schema-matching framework supports both name- and instance-based matchers. We ran COMA++ with three configurations: name matching, instance matching, and a combination of both. To emulate approaches used in cross-language ontology alignment [36, 40], we tested a variation of COMA++ where Google Translator [46]² and our automatically generated dictionary (Section 3.3.1) were used to translate attribute labels and values, respectively. The best configuration for Pt-En uses translations for both attribute names and values. For Vn-En, translating only the values provided the best results.³

3.4.2.1 Weighted Precision and Recall

Table 3.2 shows the results of the weighted evaluation measures for the alignments derived by the different approaches applied to all entity types in our datasets. Here, we show only the results

²<http://google.com/translate>

³We also experimented with different similarity thresholds and selected the values that led to the best F-measure score.

Table 3.2: Weighted Precision (P), Recall (R), and F-measure (F) for the different approaches.

Portuguese-English												
Type	WikiMatch			Bouma			COMA++			LSI		
	P	R	F	P	R	F	P	R	F	P	R	F
film	0.97	0.95	0.96	0.79	0.99	0.88	0.99	0.95	0.97	0.01	0.20	0.02
show	1.00	0.89	0.94	0.82	0.68	0.75	0.98	0.52	0.68	0.07	0.05	0.06
actor	1.00	0.52	0.68	1.00	0.24	0.39	0.70	0.52	0.60	0.15	0.26	0.19
artist	1.00	0.72	0.84	1.00	0.55	0.71	1.00	0.34	0.51	0.75	0.50	0.60
channel	0.80	0.69	0.74	1.00	0.33	0.50	0.89	0.56	0.68	0.26	0.40	0.32
company	0.86	0.87	0.87	1.00	0.53	0.69	0.95	0.70	0.81	0.67	0.74	0.71
comics ch.	0.97	0.87	0.92	0.99	0.65	0.79	0.99	0.77	0.86	0.37	0.53	0.43
album	1.00	0.93	0.96	1.00	0.69	0.82	1.00	0.77	0.87	0.56	0.48	0.52
adult actor	0.84	0.59	0.69	1.00	0.26	0.41	0.73	0.43	0.54	0.22	0.19	0.20
book	0.80	0.75	0.77	0.75	0.58	0.66	0.75	0.66	0.70	0.15	0.36	0.21
episode	0.81	0.90	0.85	0.86	0.32	0.47	1.00	0.38	0.55	0.09	0.17	0.12
writer	1.00	0.49	0.65	1.00	0.22	0.36	1.00	0.27	0.43	0.60	0.49	0.54
comics	0.92	0.65	0.76	1.00	0.13	0.23	0.91	0.45	0.61	0.00	0.00	0.00
fictional ch.	1.00	0.69	0.82	1.00	0.06	0.11	0.81	0.81	0.81	0.36	0.37	0.36
Avg	0.93	0.75	0.82	0.94	0.45	0.55	0.91	0.58	0.69	0.30	0.34	0.31

Vietnamese-English												
Type	WikiMatch			Bouma			COMA++			LSI		
	P	R	F	P	R	F	P	R	F	P	R	F
film	1.00	0.99	0.99	1.00	0.99	0.99	1.00	0.91	0.95	0.65	0.62	0.63
show	1.00	0.88	0.93	1.00	0.36	0.53	1.00	0.61	0.76	0.57	0.49	0.53
actor	1.00	0.49	0.66	1.00	0.28	0.44	1.00	0.39	0.56	0.49	0.35	0.41
artist	1.00	0.65	0.79	1.00	0.32	0.48	1.00	0.25	0.40	0.72	0.50	0.59
Avg	1.00	0.75	0.84	1.00	0.49	0.61	1.00	0.54	0.67	0.61	0.49	0.54

for the configurations that led to the highest F-measure. In Table 3.2, the last row for each language pair shows the average across all types. The highest scores for each type/metric are shown in bold.

WikiMatch obtained the highest F-measure values for almost all types and language pairs. Its recall is lower than Bouma’s for *film* in Pt-En, because it missed correct matches involving rare attributes, which occur in less than 0.5% of the infoboxes. In terms of precision, Bouma and COMA++ outperformed WikiMatch for some types. Still, considering the results averaged across all entity types, we tie in precision for Vn-En and come very close for Pt-En. By appropriately setting the thresholds, our approach can be tuned to obtain higher precision. However, since one of our goals is to improve recall for multilingual queries (see Section 3.5), where having more matches leads to the retrieval of more relevant answers, we aim to obtain a balance between recall and precision. We also study the problem of threshold stability in Section 3.4.5.

Figure 3.4 shows the results for different configurations of COMA++: name matcher (N), instance matcher (I), name+instance matcher (NI), using Google Translator for attribute names

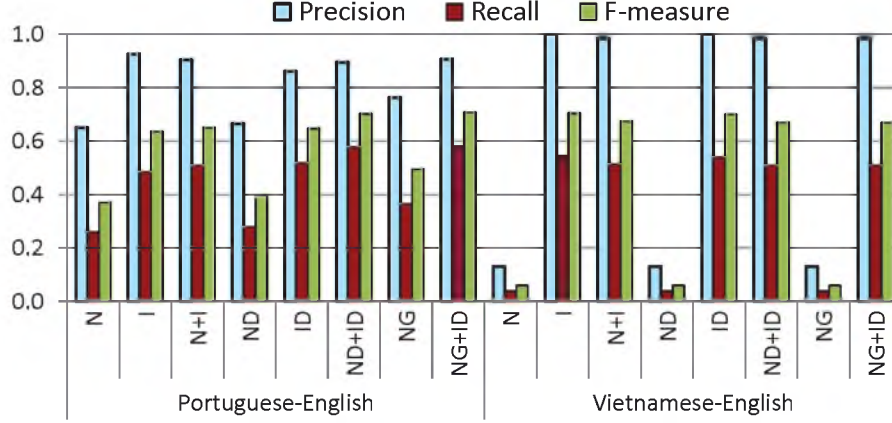


Figure 3.4: All configurations tried with COMA++

(N+G), and our automatically constructed dictionary for instances (I+D) and attribute names (N+D). We also tried different similarity thresholds (λ) from 0 to 1 with increments of 0.1⁴. We chose the configuration NG+ID for Pt-En, and ID for Vn-En and $\lambda = 0.01$ since these led to the highest F-measure. NG+ID had the best results in Pt-En because it combines information from more sources (names, instances, and translation). Note that the I configuration performs almost as well as the best configurations which use translation. While translation helps in some cases, in other cases, an incorrect translation does more harm than good. For Vn-En, the translation of attribute names was not helpful. For instance, *dien vien* was translated to *actor* instead of *starring*, and *kinh phi* was translated to *funding* instead of *budget*. When N and I matchers are combined, the N matcher returns higher similarity scores and thus takes precedence over the more reliable but lower scores of the I matcher. Therefore, NG+ID has worse results than ID only, for Vn-En. We note that even with a similarity threshold as low as 0.01, the highest recall for the best configuration of COMA++ is 0.58 for Pt-En and 0.54 for Vn-En, while for WikiMatch, at low thresholds, we obtain recall around 0.75.

WikiMatch outperforms the multilingual COMA++ configurations. This indicates that the combination of machine translation and string similarity is not effective for determining multilingual matches. This observation is also supported by the low F-measure scores for the name-based matching configuration.

⁴For each configuration, we used the mode Multiple(0,0,0) to select candidate matches as it yielded the highest F-measure.

3.4.2.2 Precision and Recall Curves

We also show the Precision/Recall curves for the different approaches. The curves were built according to the guidelines presented in Manning et al. [72]. We interpolate the precision value for a standard recall level from 0.1, 0.2 to 1.0. The interpolated precision at the j^{th} standard recall level is the maximum known precision at any recall level between j^{th} and $(j+1)^{th}$. In the precision and recall curves in Figure 3.5, we can observe that WikiMatch has higher precision and recall than LSI and COMA++. LSI can obtain high recall but very low precision.

3.4.2.3 Comparing with Consistency Learning

For WikiMatch, to assess the effectiveness of our approach for combining the features, we have built a classifier using the same features as WikiMatch (value, link, and LSI) with pseudo training data and an unsupervised consistency learning approach. To apply consistency learning, we split data into cross-validation-style folds, use one partition as pseudo-training data for predicting the matching pairs in the other folds, then shift the pseudo-training fold and repeat the prediction, and finally pick the pairs that are consistently predicted as matches over all pseudo-training folds. In order to be considered positive, the attribute pair had to be predicted as such by at least 6 of the 10 folds. We used the same three features as WikiMatch and Logistic Regression [74] as the classifier. For pseudo-training, the positive examples are automatically created by using GoogleTranslate and attribute name similarity (3-gram with different thresholds). We also use the cross-language

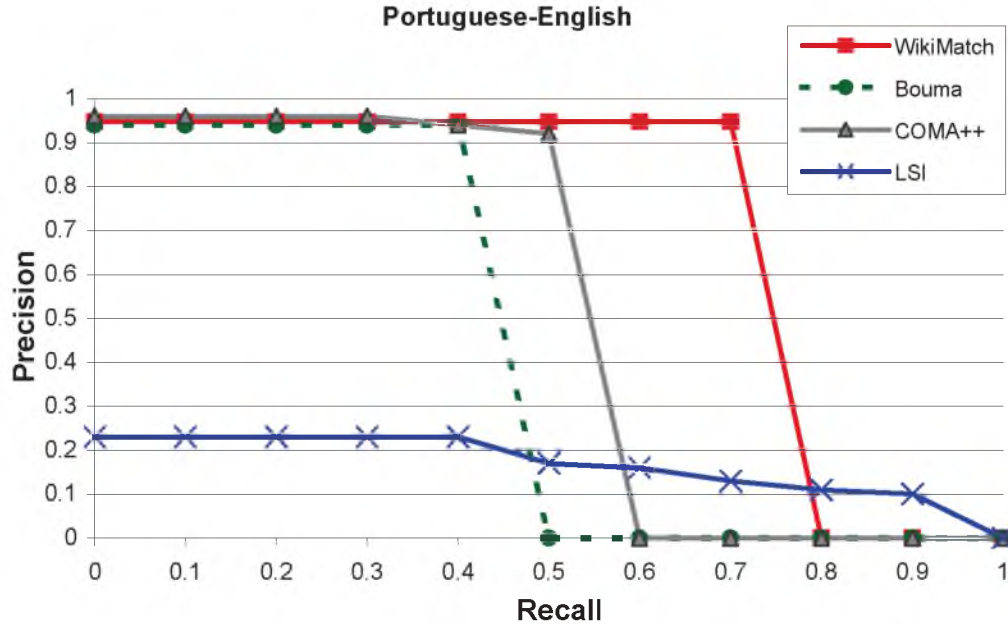


Figure 3.5: PR-Curves of different approaches in PT

links (with the same 0.6 threshold as WikiMatch) to suggest additional positive examples. Since the number of negative examples is dominant, to avoid bias, we filtered the negative examples proportionally to the number of positive examples.

As shown in Table 3.3, precision is very high, as the approach is able to consistently identify highly confident matches, but at the cost of a lower recall. The F-measure values obtained by the classifier were considerably lower than the ones obtained by WikiMatch. This indicates that combining features in a prudent way is key to obtaining high accuracy for match derivation. We should also note that creating pseudo-training data for attribute names in different languages is a nontrivial task, since it requires the translation of attribute names for different languages (which are normally not present in our automatically generated dictionary). This is challenging for a number of reasons, including: machine translation systems are either not accurate for our data or unavailable for widespread use; and some underrepresented languages (e.g., Vietnamese) lack

Table 3.3: Results by the classifier with pseudo-training and consistency learning

Portuguese-English						
Type	WikiMatch			Consist. Based		
	P	R	F	P	R	F
film	0.97	0.95	0.96	1.00	0.85	0.92
show	1.00	0.89	0.94	1.00	0.65	0.79
actor	1.00	0.52	0.68	1.00	0.52	0.68
artist	1.00	0.72	0.84	1.00	0.40	0.57
channel	0.80	0.69	0.74	1.00	0.30	0.47
company	0.86	0.87	0.87	1.00	0.72	0.84
comics ch.	0.97	0.87	0.92	1.00	0.76	0.86
album	1.00	0.93	0.96	1.00	0.40	0.57
adult actor	0.84	0.59	0.69	1.00	0.28	0.44
book	0.80	0.75	0.77	1.00	0.50	0.66
episode	0.81	0.90	0.85	1.00	0.37	0.54
writer	1.00	0.49	0.65	1.00	0.69	0.82
comics	0.92	0.65	0.76	1.00	0.27	0.43
fictional ch.	1.00	0.69	0.82	1.00	0.69	0.81
Avg	0.93	0.75	0.82	1.00	0.53	0.67
Vietnamese-English						
Type	WikiMatch			Consist. Based		
	P	R	F	P	R	F
film	1.00	0.99	0.99	1.00	0.48	0.65
show	1.00	0.88	0.93	1.00	0.23	0.38
actor	1.00	0.49	0.66	1.00	0.23	0.37
artist	1.00	0.65	0.79	1.00	0.40	0.57
Avg	1.00	0.75	0.84	1.00	0.34	0.49

translation resources. In addition, name similarity can be misleading, especially for languages of different morphologies. Mistakes in this process lead to low quality gold data, which in turn, negatively impacts the F-measure scores, and especially the recall. Without requiring training data, our approach could be scalable and applicable even to languages that are underrepresented (i.e., languages for which obtaining sufficient training samples may not be possible).

3.4.2.4 Macro-averaging

The weighting employed in the evaluation metrics in Equation 4.3 and 4.4 can be considered as *micro-averaging*. We also computed *macro-averaging* by discarding the weights and just counting distinct attribute-name pairs. The results in Table 3.4 show that WikiMatch still outperforms the other approaches.

3.4.3 Contribution of Different Components

We analyzed how much each component of WikiMatch contributes to the results by running it multiple times, and each time removing one of its components. The results, averaged over all types, are summarized in Table 3.5. WikiMatch leads to the highest F-measure values, showing that the combination of its different components is beneficial.

3.4.4 Exploring Different Alternatives for Attribute Correlation

Besides LSI, we also explore different measures for cross-language attribute correlation. The following possibilities were considered to capture the correlation between attributes a_p and a_q :

- $X1 = O_{pq}$
- $X2 = (1 + \frac{O_{pq}}{O_p})(1 + \frac{O_{pq}}{O_q})$
- $X3 = \frac{O_{pq} \cdot O_{pq}}{O_p + O_q}$

O_p and O_q are the number of occurrences of attributes a_p and a_q , respectively, and O_{pq} is the number of times they co-occur in the set of dual-language infoboxes of entity type T . Recall that the correlation score is used to order the candidate matches (Algorithm 3). Therefore, the best correlation measure for our approach is the one that leads to an ordering where the correct matches appear before the incorrect ones. We analyzed the ranking of matches produced by each of these

Table 3.4: Macro-averaging results

	WikiMatch			Bouma			COMA++			LSI		
	P	R	F	P	R	F	P	R	F	P	R	F
PT-EN	0.88	0.60	0.71	0.93	0.36	0.52	0.79	0.47	0.59	0.27	0.28	0.27
VN-EN	1.00	0.58	0.73	1.00	0.34	0.51	0.93	0.45	0.60	0.60	0.43	0.50

Table 3.5: Contribution of different components

Configuration	Portuguese-English			Vietnamese-English		
	P	R	F	P	R	F
WikiMatch	0.93	0.75	0.82	1.00	0.75	0.84
WikiMatch-ReviseUncertain	0.94	0.54	0.66	1.00	0.59	0.72
WikiMatch-IntegrateMatches	0.84	0.70	0.75	0.95	0.74	0.82
WikiMatch random	0.74	0.40	0.50	0.77	0.56	0.64
WikiMatch single step	0.39	0.89	0.52	0.56	0.88	0.64
WikiMatch-vsım	0.90	0.43	0.58	1.00	0.51	0.68
WikiMatch-lsim	0.92	0.74	0.82	0.87	0.70	0.78
WikiMatch-LSI	0.83	0.64	0.72	0.89	0.69	0.78

measures in terms of *mean average precision* (MAP) [72], which is the standard evaluation measure for ranked items in information retrieval. It is calculated as follows:

$$MAP(A) = \frac{1}{|A|} \sum_{j=1}^{|A|} \frac{1}{m_j} \sum_{k=1}^{m_j} P(R_{jk}) \quad (3.10)$$

where $|A|$ is the number of attributes in language L ; R_{jk} is the set of ranked pairs from the top result until attribute a_k ; P is the precision; and m_j is the number of correct matches for attribute j . A perfect ordering ($MAP = 1$) would place all correct matches before the first incorrect match.

MAP values for LSI and the variations of the correlation score are shown in Table 3.6. To provide a baseline for comparison, we use a random ordering for the attribute pairs. The results show that LSI provides the best ordering. Note, however, that all variations of X are superior to random ordering. The superiority of LSI can be attributed to two factors: the dimensionality reduction brought by SVD, which groups together similar infoboxes; and the fact that in addition to the co-occurrence frequency in dual-language infoboxes (which is also considered in $X1$, $X2$, and $X3$), it takes into account the occurrence pattern of the attribute pairs over the dual-language infoboxes (through the cosine distance).

3.4.5 Threshold Sensitivity

We have studied the sensitivity of WikiMatch to variations in the thresholds used in our algorithms. Figure 3.6 shows the variation of the weighted F-measure as the thresholds T_{sim} and T_{LSI}

Table 3.6: MAP for different sources of correlation

Language Pair	LSI	X1	X2	X3	Random
Portuguese-English	0.43	0.26	0.39	0.35	0.18
Vietnamese-English	0.57	0.30	0.54	0.43	0.22

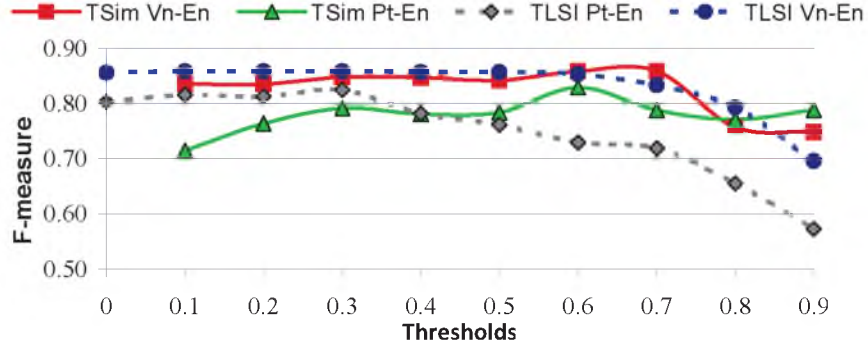


Figure 3.6: Impact of different thresholds on WikiMatch

increase. The lines show that WikiMatch is stable over a broad range of threshold values. As a general guideline, T_{LSI} should be set low since the main purpose of LSI is to sort the candidate matches, while T_{sim} should be set high as it determines the selection of the high-confidence matches. We observe a similar behavior for both language pairs: although the highest F-measure is achieved around $T_{sim} = 0.6$, the values obtained for all thresholds are comparable. The LSI score is used to sort the priority queue containing the candidate pairs. However, only attribute pairs that surpass T_{LSI} are inserted into this queue. Again, the curves for T_{LSI} are similar for both language pairs. The F-measure changes very little for T_{LSI} values between 0 and 0.6. High values of T_{LSI} reduce recall and, as a consequence, the F-measure also decreases.

3.5 Case Study: Evaluating Cross-language Queries

The usual approach to answering cross-language queries is to translate the user query into the language of the articles, and then proceed with monolingual query processing. As shown in the following experiment, our attribute correspondences can help retrieval systems in this translation process.

We ran a set of 10 c-queries in Portuguese and Vietnamese on the respective language datasets (see Table 3.7). We then translated the queries into English (as described above) and ran them over the English dataset. For each query, the top 20 answers were presented to two evaluators who were required to give each answer a score on a five-point relevance scale. The results were evaluated in terms of cumulative gain (CG) [58], which has been widely used in information retrieval. CG is the total relevance score of all answers returned by the system for a given query and it allows us to examine the usefulness, or gain, of a result set. Figure 3.7 shows the CG for Portuguese queries run over the Portuguese infoboxes (Pt), the CG for Vietnamese queries run over the Vietnamese infoboxes (Vn), and the CG for these queries translated into English run against

Table 3.7: List of c-queries used in the case study and their meaning

k Query	
1 • filme(nome=?) and ator(ocupação="político")	List all movies with an actor who is also a politician
• phim(tên=?) and diễn viên (công việc ="chính khách")	
2 • filme(nome=?) and ator(nome=?) and diretor(nome="francis ford coppola")	List all actors who worked with director Francis Ford Coppola in a movie
• phim(tên=?) and diễn viên(tên=?) and đạo diễn(tên="francis ford coppola")	
3 • filme(direção=?) and prêmio(melhor filme=?) and diretor(nascimento país de nascimento país data de nascimento="Inglaterra")	List all movies that won Best Picture Award and were directed by a director from England
• phim(đạo diễn=?) and giải thưởng(phim xuất sắc nhất=?) and đạo diễn(sinh nơi sinh="anh")	
4 • filme(receita > 10000000) and diretor(nascimento data de nascimento >=1970)	List all movies directed by a director younger than 40 (born after 1970) and that have gross revenue greater than 10 million
• phim(doanh thu thu nhập >10000000) and đạo diễn(sinh ngày sinh >=1970)	
5 • livro(nome=?) and escritor(nascimento<1975)	List all books that were written by a writer born before 1975
• sách(tên=?) and nhà văn(ngày sinh<1975)	
6 • artista(nome=?, nascimento país de nascimento país data de nascimento="França", gênero="Jazz")	List all French Jazz artists
• nghệ sĩ(tên=?, sinh nơi sinh="Pháp", thể loại="Jazz")	
7 • personagem (nome=?, criado por="Eric Kripke")	List the characters created by Eric Kripke
• nhân vật(tên=?, sáng tác="Eric Kripke")	
8 • album(nome=?, gênero = "Rock", gravado em <1980)	List the names of the albums from the genre "rock" recorded before 1980
• album(tên=?, thể loại = "Rock", ghi âm thu âm <1980)	
9 • artista(nome=?, gênero = "Rock Progressivo", nascimento data de nascimento > 1950)	List the names of artists of the genre "progressive rock" who were born after 1950
• nghệ sĩ(tên=?, thể loại = "Progressive Rock", sinh năm sinh > 1950)	
10 • companhia (sede=?, faturamento > 10 bilhões)	List the headquarters of companies with revenue greater than 10 billion
• công ty(trụ sở trụ sở chính=?, doanh thu thu nhập > 10 billion)	

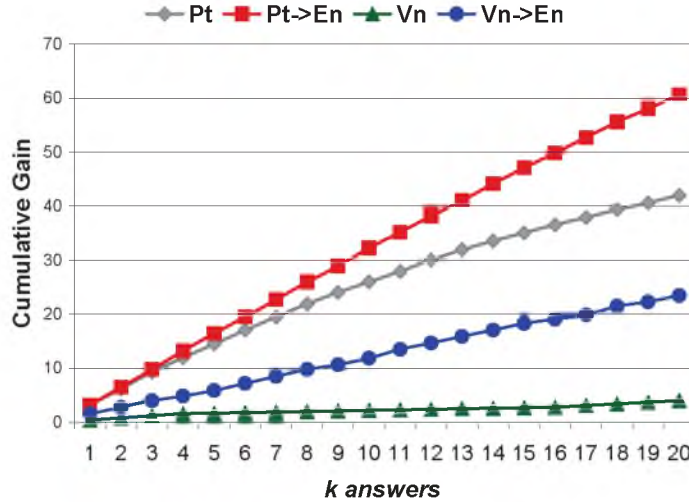


Figure 3.7: Cumulative Gain of k answers

the English infoboxes ($Pt \rightarrow En$ and $Vn \rightarrow En$). We can see that CG is always larger for the queries translated into English. This shows that our attribute correspondences help the translation and lead to the retrieval of more relevant answers. Because the English dataset covers a considerable portion of the contents both in Portuguese and Vietnamese infoboxes, it often returns many more answers.

3.6 Related Work

Cross-language matching has received a lot of attention in the information retrieval and natural language processing communities ([35, 71] and others). While their focus has been on documents represented in plain text, our work deals with structured information. More closely related to our work are recent approaches to ontology matching, schema matching, and infobox alignment.

3.6.1 Cross-Language Ontology Alignment

Fu et al. [40] and Santos et al. [36] proposed approaches that translate the labels of a source ontology using machine translation, and then apply monolingual ontology matching algorithms. The Ontology Alignment Evaluation Initiative (OAEI) [84] had a task called *very large crosslingual resources* (VLCR). VLCR consisted of matching three large ontologies, including DBpedia, WordNet, and the Dutch audiovisual archive, and made use of external resources such as hypernyms relationships from WordNet and EuroWordNet—a multilingual database of WordNet for several European languages. Although related, there are important differences between these approaches and ours. While ontologies have a well-defined and clean schema, Wikipedia infoboxes are heterogeneous and loosely defined. In addition, these works consider ontologies in isolation and do not take into account values associated with the attributes. As we have discussed in Section 3.4,

value similarity is an important information to accurately determine matches. Last, but not least, in contrast to VLCR, our approach does not rely on external resources.

3.6.2 schema-matching

The problem of matching multilingual schemas has been largely overlooked in the literature. The notable work we found on this topic aimed to identify attribute correspondences between English and Chinese schemas [101], relying on the fact that the names of attributes in Chinese schemas are usually the initials of their names in PinYin (i.e., romanization of Chinese characters). This solution not only required substantial human intervention and a manually constructed domain ontology, but it only works for Chinese and English. Although it is possible to combine traditional schema-matching approaches [88] with automatic translation (similar to [40, 36]), as shown in Section 3.4, this is not effective for matching multilingual infoboxes.

Also related to our approach are techniques for uncertain schema-matching and data integration. Gal [43] defined a class of *monotonic* schema matchers for which higher similarity scores are an indication of more precise mappings. Based on this assumption, they suggest frameworks for combining results from the same or different matchers. However, due to the heterogeneity across infoboxes, this assumption does not hold in our scenario: matches with high similarity scores are not necessarily accurate. To this hypothesis, we have experimented with different similarity thresholds for COMA++, and for higher thresholds, we have observed a drop in *both* precision and recall.

3.6.3 Cross-Language Infobox Alignment

Adar et al. [2] proposed Ziggurat, a system that uses a self-supervised classifier to identify cross-language infobox alignments. The classifier uses 26 features, including equality between attributes and values and n-gram similarity. To train the classifier, Adar et al. applied heuristics to select 20K positive and 40K negative alignment examples. Through a 10-fold cross-validation experiment with English, German, French, and Spanish, they report having achieved 90.7% accuracy. Bouma et al. [20] designed an alignment strategy for English and Dutch which relies on matching attribute-value pairs: values v_E and v_D are considered matches if they are identical or if there is a cross-language link between articles corresponding v_E and v_D . A manual evaluation of 117 alignments found only two errors. Although there has not been a direct comparison between these two approaches, Bouma et al. state that their approach would lead to a lower recall. However, the superior results obtained by Ziggurat rely on the availability of a large training set, which limits its applicability for languages that have limited number of infoboxes: training is required for each different domain and language pair considered and the approach is likely to be effective only for domains and languages that have a large set of representatives. Adar et al. acknowledge

that because their approach relies heavily on syntactic similarity (it uses n-grams), it is limited to languages that have similar roots. In contrast, `WikiMatch` is automated—requiring no training, and it can be used to create alignments for languages that are not syntactically similar, such as, for example, Vietnamese and English. Nonetheless, we would have liked to compare `Ziggurat` against our approach, in particular, for the Pt-En language pair. Unfortunately, we were not able to obtain the code or the datasets described in Adar et al. [2].

3.6.4 Cross-Language Concept Alignment

While `WikiMatch` focuses on finding synonyms for infobox attributes, UWN [29] focuses on constructing a multilingual lexical knowledge-base using WordNet to make the semantic connections between concepts in different languages explicit, and MENTA [30] focuses on finding synonyms for categories by exploiting the concept hierarchy and their cross-language links. Because the cross-language links for attribute names are very limited today, these approaches are not applicable to 2012 Wikipedia infoboxes.

3.7 Conclusion

In this chapter, we proposed `WikiMatch`, a new approach for aligning Wikipedia infobox schemas in different languages that requires no training and is effective for languages with different morphologies. Furthermore, it does not require external sources such as dictionaries or machine translation systems. `WikiMatch` explores different sources of similarity and combines them in a systematic manner. By prioritizing high-confidence correspondences, it is able to minimize error propagation and achieve a good balance between recall and precision. Our experimental analysis showed that `WikiMatch` outperforms state-of-the-art approaches for cross-language information retrieval, schema-matching, and multilingual attribute alignment; and that it is effective for types that have high cross-language heterogeneity and few data instances. We also presented a case study that demonstrates the benefits of the correspondences discovered by our approach in answering multilingual queries over Wikipedia. By using the derived correspondences, we can translate queries posed in underrepresented languages into English, and as a result, return a larger number of relevant answers.

We should note that not all correct attribute pairs co-occur in any dual-language infobox in the dataset and thus, they will not be found. For example, no dual-language (Pt-En) infobox contains the attributes `premios` and `awards` even though they are synonyms. Like other approaches, `WikiMatch` is not able to identify such matches since all similarity measures return low scores. However, these are rare matches, which as we see from the results, do not significantly compromise recall. Another limitation of `WikiMatch` is that, currently, it does not support languages that do not use alphabetical

characters. In order to build a local language dictionary, we need to combine with approaches such as cross-language concept alignment [29], and ontology alignment [94, 36, 40].

We used the entity types with their schemas that are discovered by `WIClust` (see Chapter 5) to obtain types. Possible alternatives are from DBpedia [17] or from YAGO [95]. However, because of manual effort, today the coverage of DBpedia is limited; particularly, there are no defined types for Portuguese and Vietnamese Wikipedia. In addition, the heuristics to extract plural conceptual terms in YAGO is language-dependent and does not work for languages like Vietnamese because the quantifiers indicating plural nouns were omitted as we examine the Vietnamese category names.

CHAPTER 4

A GENERAL PRUDENT SCHEMA-MATCHING FRAMEWORK

We have presented the problem of matching form interfaces in Chapter 2 and presented the problem of matching multilingual Wikipedia infoboxes in Chapter 3. Both of these problems share some common traits and must work on a large number of heterogeneous and noisy schemas. Although there are multiple sources of similarity, there is no single best way to combine the similarities. In order to have at least one high-precision matcher, we used constraints to obtain high-confidence matches first. Given the explosion of structured Web data, we proposed an approach that is able to automatically derive accurate matches for a large dataset, without relying on manual preprocessing or simplification, while also remaining unaffected by rare attributes and resistant to noise inherited from previous steps in the integration process such as the label extraction process or resilient to the problem of schema drift from the mass community contribution. As a further step forward from these problems, we generalize the prudent schema-matching framework – `PruSM`, which combines latent and apparent features to automatically find matches for a large number of schemata. We show how to customize `PruSM` for different scenarios.

4.1 The PruSM Matching Framework

The general `PruSM` matching framework is shown in Figure 4.1. From the Schema Repository, all attributes that have the same name are grouped together and their domain values are aggregated. Attributes with a frequency higher than a (low) threshold are considered as frequent attributes and the remainder are infrequent attributes. Only frequent attributes are considered as the input for the Matching Discovery, which combines different similarities in a prudent fashion to discover certain matches. Uncertain matches will be revised and added to the set of matches. The Matching Growth component finds additional matches for rare attributes. `PruSM` starts from certain matches to incrementally extend the result. Notable features of `PruSM` include requiring neither preprocessing (e.g., syntactic merging) nor the use of an external thesaurus. `PruSM` is robust to rare attributes and resilient to inherent noise from previous automatic processes or the crowdsourcing contribution.

Below, we will present principle parts of `PruSM`, including the prudent matcher and the matching algorithm. We then show how to customize `PruSM` for different scenarios.

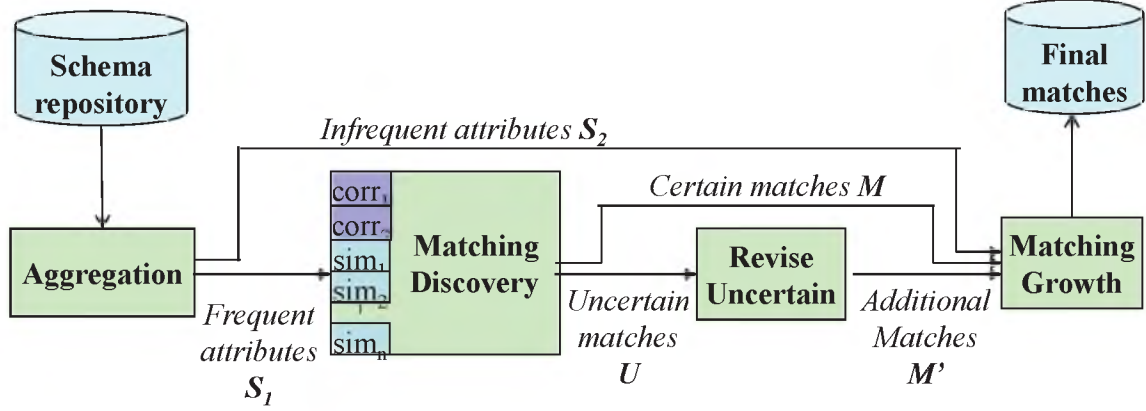


Figure 4.1: PruSM matching framework

4.1.1 Prudent Matcher

The goal of the prudent matcher is to identify matches with high confidence. Our motivation for the prudent matcher comes from the fact that, since large-scale Web data are very heterogeneous, the assumption of monotonic matchings [43] no longer holds (i.e., a higher similarity score is not an indication of an accurate mapping). Therefore, it is very useful to have a simple yet comprehensive matcher that is able to find confident matches and avoid propagation errors as early as possible. We propose a prudent matcher that incorporates both apparent and latent features. In contrast to pair-wised approaches, the similarity measures are defined on sets of aggregated elements, such as attribute correlations, value similarity, cross-link similarity, and so on. By analyzing the attribute co-occurrence patterns from a large number of schemas, we can leverage an implicit source of similarity information: attribute correlation. In particular, we use X/Y correlation for matching Web-forms and LSI for matching multilingual infoboxes. Details about these correlations were discussed in previous chapters. The intuition is that similar attributes rarely co-occur in the same schema and have a high negative correlation (*NC*). For attributes in the same language, a negative correlation score of 1 means they never co-occur in a dual-language infobox. Consequently, they are likely to be intralanguage synonyms. In contrast, for attributes in different languages, for the dual-language infoboxes, a correlation score of 1 means that they co-occur in every dual-language infobox and thus, have a good chance of being cross-language synonyms.

The correlation alone is not sufficient as it would be artificially high in some cases, especially in the presence of rare attributes or incomparable schemata [53, 79]. In addition to correlation, we also use other supporting similarities available for forms and infoboxes such as label similarity, value similarity, and cross-link similarity.

Although the features are obtained at the level of set of elements, using them in isolation is

still insufficient and can lead to error propagation. Given many aspects of similarity, as shown in previous chapters, we argue that there is no single best way to combine similarity for different domains, or even pairs of schemata. However, we observe that correlation can be effective if prudently combined and reinforced with additional evidence, such as strong label similarity and value similarity for Web-forms or value similarity and link similarity for Wikipedia infoboxes. Combining different features can potentially compensate for each matcher's weaknesses.

The prudent matcher is defined as follow:

$$\Omega_M(a_x, a_y) = \text{sim}(a_x, a_y) \quad (4.1)$$

$$C_i(M_i, T_i) = \text{constraint}[\text{sim}_i > T_i] \quad (4.2)$$

$$\Omega_{M^*}(a_x, a_y) = \text{CORR}(a_x, a_y) \quad (4.3)$$

$$C_i^*(M_i^*, T_i) = \text{constraint}[\text{CORR}_i > T_i] \quad (4.4)$$

$$\text{PruC}(a_x, a_y) = \text{Conjunction}(C_l, C_k^*) \quad (4.5)$$

$$\Omega_{\text{PruM}}(a_x, a_y) = \begin{cases} \text{CORR}(a_x, a_y) & \text{if } \text{PruC}(a_x, a_y) \text{ is satisfied} \\ 0 & \text{otherwise} \end{cases} \quad (4.6)$$

$$\sigma_i^{\text{PruM}} = \text{argmax}_{\sigma \in \Sigma_T} \Omega(\text{PruM}(A)) | \sigma_1, \sigma_2, \dots, \sigma_{i-1} \quad (4.7)$$

Equations 4.1 to 4.4 define the similarity measure Ω for a matcher M and the similarity constraint C for every two attributes (Ω^* and C^* denote the correlation and constraint for the correlation matcher). The prudent constraint PruC (Equation 4.5) is a conjunction of similarity constraints, including the correlation constraint. The prudent constraint ensures that even if two attributes have a high correlation score, they do not provide a good match unless additional evidence is available. If the prudent constraint is satisfied, the prudent matcher PruM returns the correlation score (Equation 4.6). The i^{th} -prudent match σ_i is the match with i^{th} -highest score returned from the prudent matcher, given the previous matches from σ_1 to σ_{i-1} (Equation 4.7). In particular, this match will be integrated into the set of previous matches by checking the correlation constraints among these attributes (details are in the `IntegrateMatches` algorithm in Chapters 2 and 3). As illustrated in the following example, the orderings returned from the prudent matcher are important to the match derivation process.

Example 12. Let attributes A, D, E, and F be correct matches as illustrated in Figure 4.2. Supposing $NC(A, B) > NC(A, D)$, the correlation matcher will consider attribute A to match with attribute B and then match with attribute C. Because attribute D is not correlated with attribute B, attribute A will never be matched with attributes D and E and F. Therefore, if a

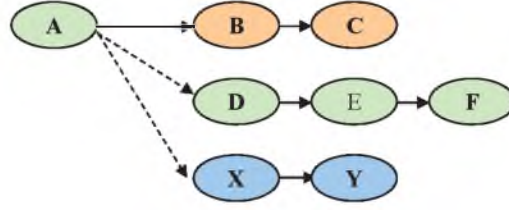


Figure 4.2: No validation can lead to incorrect matchings and consequent errors

bad decision is made in an early step, it may not be corrected and may negatively affect the following steps. By identifying high confident matches first, it is possible to avoid a potentially large number of incorrect matches. ■

Equations 4.8 to 4.9 describe the properties of the prudent matcher, which is an ensemble matcher that gives a consensus ordering from the orderings of individual matchers.

$$\sigma_k^{PruM} \preceq \sigma_{k+1}^{PruM} \quad (4.8)$$

$$\sigma_k^{M_i} \preceq \sigma_k^{PruM} \quad (4.9)$$

First of all, because the correlation is corroborated by other similarities, the k^{th} -prudent match derived by *PruM* is better than the $(k+1)^{th}$ -prudent match (Equation 4.8). Secondly, due to the reinforcement of multiple matchers, the ordering returned from the prudent matcher is better than the ordering from the individual matcher; e.g., the k^{th} best match derived by *PruM* is better than the k^{th} best match derived by an individual matcher M (Equation 4.9). Because of the previous properties, the precision of the prudent matcher is often higher than each individual matcher and other trivial combinations of them. Being able to derive certain matches, the prudent matcher is suitable and efficient for the two-step approach where we obtain high precision first and improve recall later.

4.1.2 Matching Algorithm

Let A denote the set of attributes with the same name and P denote the set of these attribute pairs. Because the relative ordering of the correlation score is important, the attribute pairs in P are sorted in *decreasing order* of the correlation score to ensure the most certain matches are processed first (line 9 Algorithm 5) to avoid the early selection of incorrect matches, which can lead to error propagation in future matches.

Only candidate correspondences which satisfy the constraint are considered as input to the algorithm *IntegrateMatches*, which decides whether it will be integrated into an existing match, originate a new one, or be ignored. *IntegrateMatches* outputs a set of matches M , where each

Algorithm 5 PruSM

```

1: Input: Set of attributes for an entity type  $T$  or a domain  $D$ 
2: Output: Set of matches  $M$ 
3: begin
4:  $M \leftarrow \emptyset, P \leftarrow \emptyset$ 
   /* $P$  denotes the attribute pair*/
5: for each pair  $\langle a_p, a_q \rangle$  such that  $a_p, a_q \in A$  do
6:   Compute  $vsim, lsim, NC$ 
7:    $P \leftarrow P \cup (\langle a_p, a_q \rangle, m_1, m_2, \dots, m_k^*)$ 
8: while  $P \neq \emptyset$  do
9:   Choose pair  $\langle a_p, a_q \rangle$  with the highest  $NC$  score
10:  if  $PruC(a_p, a_q)$  then
11:     $M \leftarrow \text{IntegrateMatches}(\langle a_p, a_q \rangle, M)$ 
12:  else
13:     $U \leftarrow \langle a_p, a_q \rangle$  /*buffering uncertain matches*/
14:    Remove  $\langle a_p, a_q \rangle$  from  $P$ 
15:     $U' \leftarrow \text{ReviseUncertain}(U)$ 
16:  for each  $u \in U'$  do
17:     $M \leftarrow \text{IntegrateMatches}(u, M)$ 
18: end

```

match includes a set of corresponding attributes. `IntegrateMatches` takes advantage of the correlations among attributes to determine how to integrate the new correspondence into the set of existing matches. Details of the algorithm are presented in Chapters 2 and 3. The idea is to test for negative correlations between all attributes of a match to see whether it is possible to integrate the attributes in question into the existing matches.

4.2 Customize PruSM for Different Case Studies

In this section, we present how to customize the PruSM framework (Figure 4.1) for different scenarios and summarize our experimental results.

4.2.1 Customizing PruSM

4.2.1.1 Customize the Correlation Measure

Although there are several correlation measures (e.g., Jaccard, Jini index, Laplace, Kappa), none is universally good [83]. In particular, we use the negative correlation NC for matching Web-forms and LSI for matching multilingual infoboxes. The intuition is that similar attributes rarely co-occur and have a high negative correlation. For multilingual infoboxes, we consider the dual-language infobox schema which contains the union of the attributes in their schemas. For a given language, the same intuition holds for attributes in an infobox—synonyms should not appear together. However, for identifying cross-language correspondences, the opposite is true: if we

combine the attribute names for corresponding infoboxes across languages in the dual-language infobox schema, cross-language synonyms are likely to co-occur.

4.2.1.2 Customize the Supporting Similarities

Besides the attribute correlation, we need to leverage information available in Web-forms or in the infoboxes to validate the correlation. In particular, we leverage the label similarity and domain similarity for Web-forms and leverage the value similarity and the link structure similarity. Details about these similarities were provided in Chapters 2 and 4.

4.2.1.3 Customize the Prudent Matcher

The similarities for a pair of attributes a_p, a_q in FormMatch are combined according to the constraint: $X(a_p, a_q) > T_{Matching_score}$ AND $[dsim(a_p, a_q) > T_{dsim}$ OR $lsim(a_p, a_q) > T_{lsim}]$. Similarly, the similarities for a pair of attributes a_p, a_q in WikiMatch are combined according to the constraint: if $LSI(a_p, a_q) > T_{LSI}$ AND $\max(vsim(a_p, a_q), lsim(a_p, a_q)) > T_{sim}$, then $\langle a_p, a_q \rangle$ is a certain candidate correspondence. The intuition is that two attributes form a certain correspondence, if they are correlated and this is corroborated by at least one of the other similarity measures.

4.2.1.4 Customize the Matching Components

Figure 4.1 shows the components of the PruSM matching framework. We can customize it for different scenarios. For example, by taking advantage of certain matches to validate uncertain matches (Revise Uncertain) in WikiMatch, or by incrementally finding additional matches for infrequent attributes (Matching Growth) in FormMatch, we can improve the final recall.

4.2.2 Summary of Results

4.2.2.1 Outperform Existing Approaches

As shown in previous chapters, PruSM outperforms the other approaches. In the case of Web-forms, it outperforms the state-of-the-art schema-matching approaches (Chapter 2). In the case of multilingual Wikipedia infoboxes, PruSM also outperforms the other matching approaches like COMA++, Bouma, and LSI [9, 20, 31] (Chapter 4). The superiority of PruSM comes from the reinforcement of correlation with other similarity information.

4.2.2.2 Different Combining Strategies

We evaluate the effectiveness of the prudent matcher by comparing it against other matchers, including single matchers (correlation, lsim, dsim); combination of different matchers like linear combination; or choosing the maximum value among them. As shown in Chapter 2, the prudent

matcher obtains substantially higher precision than the others, which is the most important goal in the Matching Discovery step.

For WikiMatch, to assess the effectiveness of our approach for combining the features, we have built a classifier using the same features as WikiMatch (value, link, and LSI) with pseudo-training data and an unsupervised consistency learning approach. The F-measure values obtained by the classifier were considerably lower than the ones obtained by WikiMatch. This indicates that combining features in a prudent way is key to obtaining high accuracy for match derivation.

4.2.2.3 Contribution of Different Features

The contribution of different features varies for different domains. Figure 4.3 shows the contribution of different features by showing the reduction in the final F1 if excluding that feature. In the Auto domain, domain values are prevalent and thus more reliable and important. In the Airfare domain, domain values are ambiguous for reciprocal attributes and thus mislead the matching process. The Book domain is more heterogeneous: correlation signal is weak and therefore needs information from both label and domain values.

Similarly, the contribution of different similarities varies for different languages (Figure 4.4). Cross-link similarity is sufficient and thus is an effective supporting similarity for Portuguese to English matching. On the other hand, cross-links are scarcer and thus, value similarity is more important for Vietnamese to English matching. Similarly, correlation plays an important role for Portuguese because it can benefit from a larger number of infoboxes.

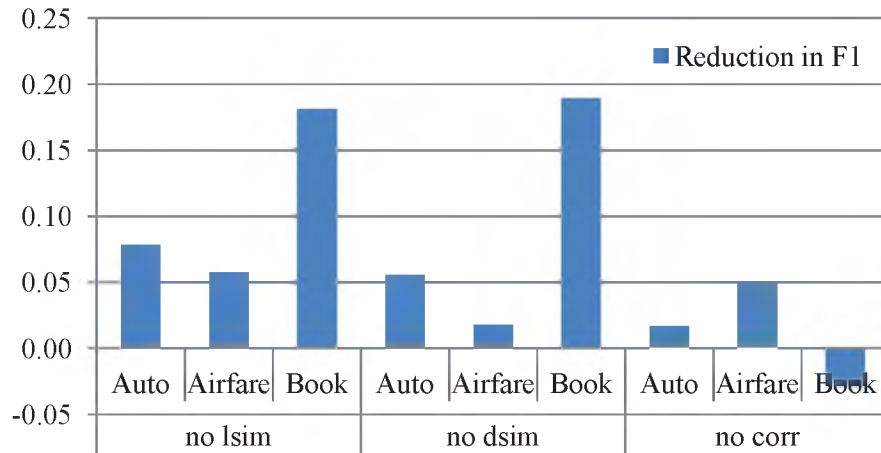


Figure 4.3: Contribution of different features in FormMatch

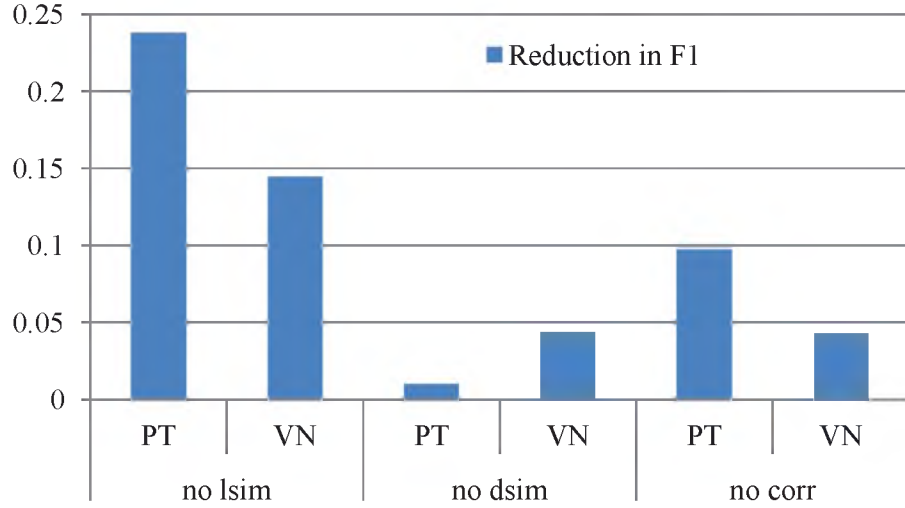


Figure 4.4: Contribution of different features in WikiMatch

4.2.2.4 Threshold Sensitivity

We studied the threshold sensitivity of PruSM for the case studies. Figure 4.5 shows the sensitivity of the Matching Discovery to different correlation thresholds, obtained by using the correlation matcher and by using the prudent matcher. In particular, the correlation matcher can obtain high precision only when using very high correlation thresholds. However, the obtained recall is lower. On the other hand, the Prudent Matcher is very effective and its accuracy is robust to a wide range of correlation thresholds, including very low correlation thresholds.

Similarly, the LSI by itself is not precise, but when combined with other similarities, it is robust over different domains. Figure 4.6 shows that WikiMatch obtains high F-measure for a broad range of threshold values.

4.3 Conclusion and Discussion

We have introduced the new PruSM matching framework. In contrast to pair-wised approaches, PruSM combines the similarities for sets of elements to find consensus terms for the whole collection. Given the Web heterogeneity, it is useful to define constraints to have a high-precision matcher. By prioritizing confident matches first to avoid error propagation and then revising uncertain matches to increase recall, PruSM is effective for matching large collections of heterogeneous and noisy schemata. PruSM has been applied to distinct problems of matching form interfaces and matching multilingual Wikipedia infoboxes. Our experiments show that PruSM can enable on-the-fly and automatic integration of large collections of structured Web data. As future work, we would like to integrate PruSM with more matchers, customize and test it on a great variety of

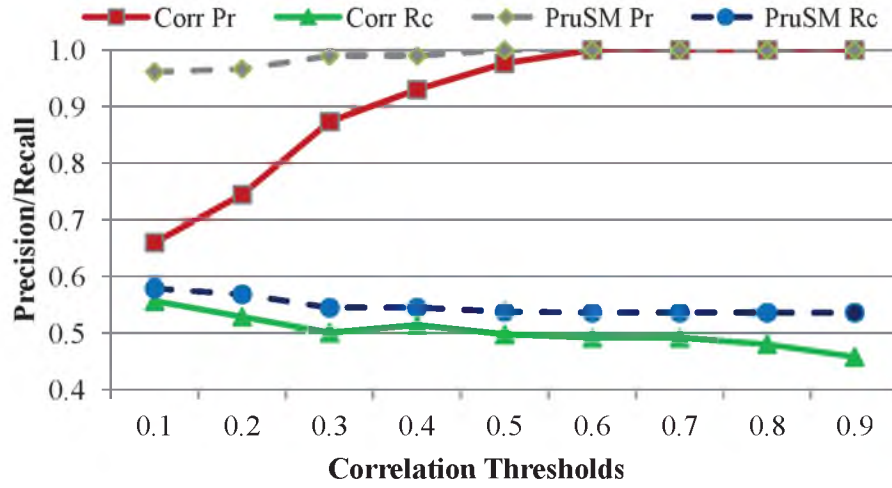


Figure 4.5: Threshold sensitivity of FormMatch

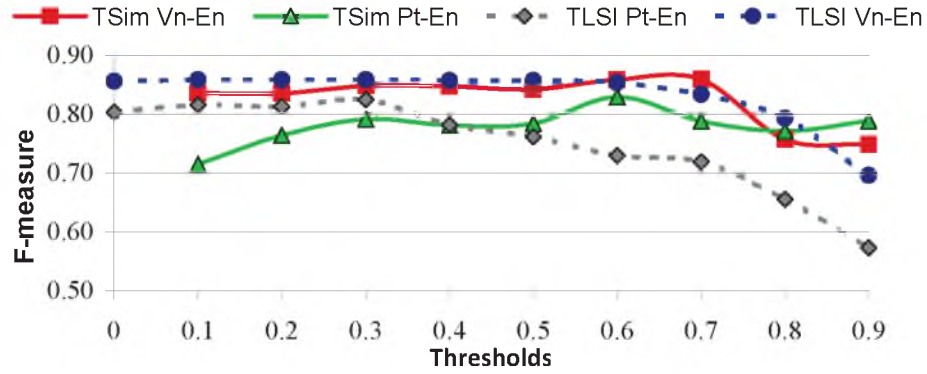


Figure 4.6: Impact of different thresholds on WikiMatch

scenarios and datasets. We also plan to make the PruSM framework with different configurable components available online.

CHAPTER 5

ORGANIZING WIKIPEDIA BY CLUSTERING INFOBOXES

5.1 Introduction

The availability of structured information in the infoboxes makes it possible to answer complex queries such as: *Find the titles and years of movies directed by James Cameron that grossed over 100 million dollars and whose stars were born in England.* To answer these structured queries, systems have been proposed [61, 78] which model each Wikipedia document as an entity with a type, a set of associated attributes (or schema), and relationships connecting it to other entities. For example, the document *Avatar*¹, which satisfies the query above, can be represented as an entity of the type *Movie*, with attributes *title*, *year*, *gross revenue*, etc. It is connected to the director *James Cameron* via a *directed by* relationship, and it is connected to the actor *Sam Worthington* who was born in *England* via a *starring* relationship.

However, the mass collaboration approach that has made such queries possible also creates significant challenges. Even though authors are encouraged to provide some structure, use infobox templates, and select appropriate categories, these guidelines are not always followed. This leads to inconsistencies, including the duplication of templates, schema drift, and data heterogeneity, which in turn make it hard to query the Wikipedia information [11, 8, 105].

Approaches have been proposed to deal with this heterogeneity by using *infobox template names* and document *categories*. An infobox² can be associated with a *template name* which corresponds to a predefined *entity type*. An entity type may encompass several different infobox templates. For example, the entity type *Film* is associated with template names *Infobox Film*, *Infobox Movie*, *Television Film Infobox*, *TV film*, *James Bond film infobox*, *Infobox Chinese film*, and *Infobox Korean film*. In DBpedia [17], the 350 most commonly used infobox templates were manually mapped to an ontology consisting of 170 classes. YAGO [95] uses language-dependent heuristics to extract concept names from Wikipedia *categories* and maps them to concepts in WordNet. KOG [105] also uses WordNet and unifies entities that have the *same* canonical infobox template names. However,

¹[http://en.wikipedia.org/wiki/Avatar_\(2009_film\)](http://en.wikipedia.org/wiki/Avatar_(2009_film))

²We use the terms entity and infobox interchangeably.

due to inconsistencies, sparseness, and noise in categories and in infobox template names, these approaches are error prone and require substantial human intervention.

While an infobox can be associated with a *template name* that corresponds to a predefined *entity type*, template names are not always a reliable source for determining entity types. Since there is no central authority, Wikipedia editors can freely create templates and associate template names to infoboxes. Thus, several names can be used for the same entity type. For example, the entity type *Film* is associated with template names `Infobox Film`, `Infobox Movie`, `Television Film Infobox`, `TV film`, `James Bond film`, `Chinese film`, `infobox Korean film`, etc. Editors also use templates with *generic* names, e.g., `wikitable`, `infobox`, or very *specific* ones, e.g., `SinCityCharacter` instead of `Character`, as illustrated in Figure 5.1. Furthermore, while some templates have descriptive names, others contain only abbreviations (e.g., `VG`, `MFL`).

The category name in Wikipedia is another source for inferring entity type. While there are many *categories* such as conceptual category, administrative category, relational category, and thematic category, only conceptual category is used to extract types [95]. Because categories are often folksonomic, conceptual categories are also an unreliable source for inferring the type of an entity. Consider, for example, Figure 5.2 which shows an infobox for a movie and its associated categories. Given these categories, a system like Yago would assign to the infobox concepts like *film*, *winner*, *olympics*, *culture*, *university*, and *sport*. However, only *film* corresponds to the entity described in the infobox. The problem is compounded because users also make mistakes in assigning documents

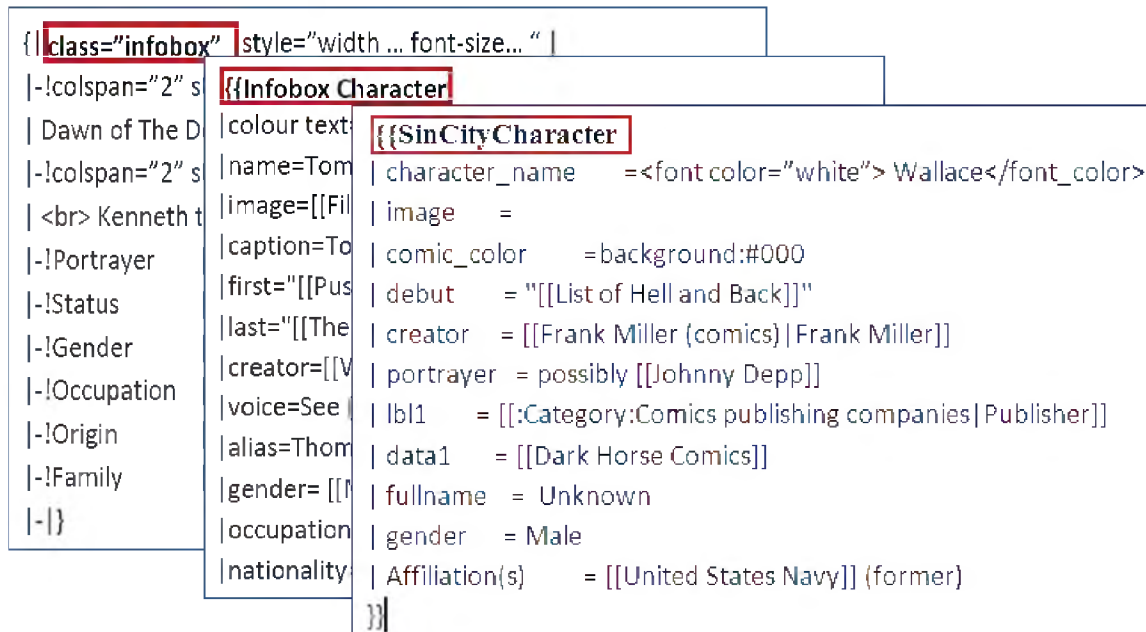


Figure 5.1: Different template names are used for the type Character

Directed by	Hugh Hudson	Categories: 1981 <u>films</u> <u>Works</u> about Gilbert and Sullivan 1924 Summer <u>Olympics</u> Olympic <u>films</u> English-language <u>films</u> French-language <u>films</u> Best Foreign Language Film Golden Globe <u>winners</u> Best Picture Academy Award <u>winners</u> <u>Films</u> whose writer won the Best Original Screenplay Academy Award Best Original Music Score Academy Award <u>winners</u> <u>Cambridge</u> in fiction <u>University</u> of Cambridge in fiction <u>Culture</u> of the University of Cambridge <u>Sport</u> at the University of Cambridge <u>Films</u> about Jews and Judaism Directorial debut <u>films</u> <u>Films</u> directed by Hugh Hudson <u>Films</u> about competitions ...
Produced by	David Puttnam	
	Dodi Fayed (executive producer)	
Written by	Colin Welland	
Starring	Ben Cross	
	Ian Charleson	
	Nigel Havers	
Music by	Vangelis	
Cinematography	David Watkin	
Editing by	Terry Rawlings	
Distributed by	20th Century Fox	
Release date(s)	30 March 1981	
Running time	118 minutes	
Country	United Kingdom	
Language	English	
Budget	\$5.5 million	
Box office	\$58.9 million (USA)	

(a)

(b)

Figure 5.2: Example of infobox and category names of a Movie

to categories and do not always provide complete information.

To get a better sense for the heterogeneity in template names and categories, consider Table 5.1. This table shows, for different entity types in different domains and languages, the coverage of the most frequent template name and conceptual term in the categories (shown in bold), and the total number of template names and categories (within parenthesis). For example, in the Portuguese Movie domain (PT_Movie), there are 12 distinct template names for *actor*; the template `Info ator` covers only 52% of the actor entities, while the category `atores` covers 85% of the actors. Given the dynamic nature of Wikipedia, with new entities, categories, and templates being created constantly, a systematic approach is needed to discover the correspondences among the template names and categories. While manual curation can be effective, it is not only costly, but as this example shows, may lead to limited coverage.

We propose to use infoboxes as a source of *type* information. The infobox for an entity provides a succinct (and structured) description which consists of a set of attributes and values that represent the entity. We posit that this structure provides a reliable means to identify the entity type, and frame the problem of entity type discovery as *clustering* infobox schemata. However, deriving high-quality clusters is challenging. First, the number of clusters that need to be generated is not known a priori. Second, the undisciplined process of infobox creation leads to a wide variation in

Table 5.1: Heterogeneity of template and category names

Entity type (Domain)	Infobox Templates	Coverage (#Templates)	Conceptual Terms in the Catagories	Coverage (#Categories)
actor (EN_Movie)	Infobox Actor , Infobox_celebrity, Infobox actor voice, Infobox Chinese language singer and actor, Infobox Korean name/Actor, Infobox	0.78 (13)	actors , actress, directors, producers, people, singers...	0.76 (10,844)
book (EN_Book)	Infobox book , Infobox short story, Infobox SW Books, Infobox Novel Series, infobox original English manga, infobox comics story arc, infobox Hymnal, Infobox Medieval text...	0.86 (22)	books , novels, stories, works, fiction, debuts, fantasy...	0.69 (4,362)
software (EN_Comp)	Infobox Software , Infobox_Software2, Infobox information appliance, Infobox Non-profit, Infobox Online music service, Infobox VG Online Service, Infobox Website...	0.85 (21)	software , tools, browsers, services, clients, engines, simulators...	0.78 (1,606)
ator (PT_Movie)	Info ator , infobox actor, info/biografia, infobox ator, info/bio adulto...	0.52 (12)	atores , atrizes, cineastas...	0.85 (1,337)
diễn viên (VN_Movie)	Infobox actor , thông tin diễn viên, infobox chinese actor and singer...	0.72 (6)	diễn viên , đạo diễn, nghệ sỹ ...	0.95 (550)

entity schemata. Besides schema heterogeneity being commonplace, even within a single entity type, there are many optional (i.e., not present in all entities of a given type) and rare attributes (i.e., present in very few instances of a given type) [105, 11]. Furthermore, the infoboxes present a *long tail* distribution for types (and schemata)—a large number of types have few associated instances and few types are associated with many instances [105, 11] (see Figure 5.3). This makes it difficult to cluster the corresponding entities appropriately, in particular, for the low-frequency templates. Last, but not least, because some attributes belong to different entity types, they can mislead the clustering algorithm to group irrelevant types.

To address these problems, we have designed a new clustering strategy, which we call **WIClust** (Clustering Wikipedia Infoboxes). **WIClust** receives as input a set of infoboxes and outputs a set of entity types. In the absence of well-defined types, it uses the correlation among infobox attributes to identify important attribute sets which, in turn, are used as the basis for clustering (structurally) similar infobox schemata. As we describe below, identifying these important sets as a first step allows **WIClust** to derive types accurately, handle ambiguous attributes, and incrementally add rare attributes, increasing the coverage of the schemata without significant loss in accuracy.

WIClust also takes advantage of the links between infoboxes to refine types. In particular, it groups together entities that have similar link structure but whose attributes may be distinct (e.g., different kinds of Person) and puts them into classes that are potentially semantically meaningful. For example, in Wikipedia, the template Actor is used not only for actors, but also for other kinds of entities like director and producer. **WIClust** is able to derive refined types for such entities.

Besides producing a set of entity types, we identify attribute synonyms as well as relationships

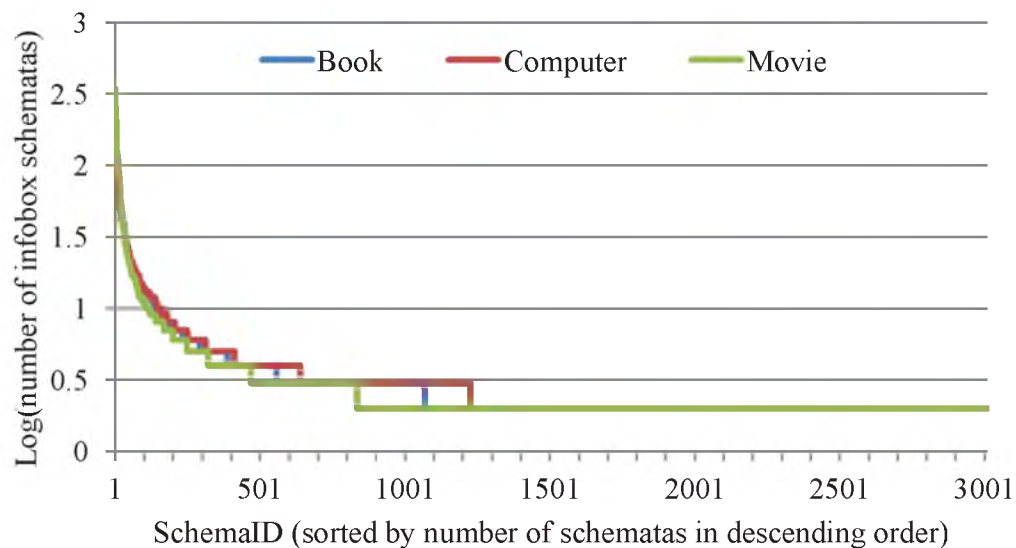


Figure 5.3: The long tail schema histogram

among entities. The synonyms not only improve the quality of derived clusters, but also help during querying, enabling a query engine to perform automatic relaxation and avoid missing relevant answers. Entity relationships, on the other hand, can be used in query reformulation to support declarative interfaces, which relieve users from having to specify the details of how to connect different pieces of information [78].

5.2 Problem Definition and Solution Overview

5.2.1 Problem Definition

Given a set of infoboxes as input, our goal is to identify a set of entity types and corresponding schemata that represent these infoboxes, as well as the relationships between the types.

Before giving an overview of our approach, we first define some concepts and terminology.

5.2.2 Terminology and Definitions

An *infobox* $ib = \{ \langle a_i, v_i \rangle \mid i = 1..m \}$ consists of a set of attribute-value pairs that describe important information about an entity. The *schema* of an infobox ib is the set of its attributes and we denote it by $S_{ib} = \{a_i \mid i = 1..m\}$. An infobox corresponds to a predefined *entity type*. For example, the schema of the infobox in Figure 5.2(a) is associated with the entity type *Movie* and includes the attribute set $\{\text{Directed by, Produced by, Written by, Starring, } \dots, \text{Box office}\}$. The schema S_T of an entity type T is the union of all attributes in infoboxes (instances) associated with T .

The value of an attribute in an infobox may contain one or more hyperlinks that point from infobox of type T_i to infobox of type T_j . We represent each hyperlink h_k by a tuple (ib_i, p_k, ib_j) , where $p_k = a_{ik}$ is the *link predicate* between infoboxes ib_i and ib_j . The presence of such a link may indicate that there is a *relationship* r_p between T_i and T_j . For example, *Starring* is a relationship between entity type *Movie* and *Actor*.

In the entity discovery process, we create attribute and entity clusters. An *attribute cluster*, corresponding to an entity type T and denoted by A_T , consists of a set of important attributes of entity type T discovered by our algorithm. A *entity cluster*, denoted by E_T , is a set of entities (infoboxes) that belong to entity type T .

5.2.3 Solution Overview

To discover the entity clusters, we apply a three-step approach, illustrated in Figure 5.4. Given a set of infoboxes, the *Attribute Clustering* step leverages the correlations among attributes to return a set of *attribute clusters* \hat{A} . Each attribute cluster comprises important attributes of an entity type T . For example, the attribute cluster discovered for the entity type *Movie* includes the attribute set $\{\text{Directed by, Produced by, Written by, Starring}\}$. In the *Infobox Grouping* step, each infobox

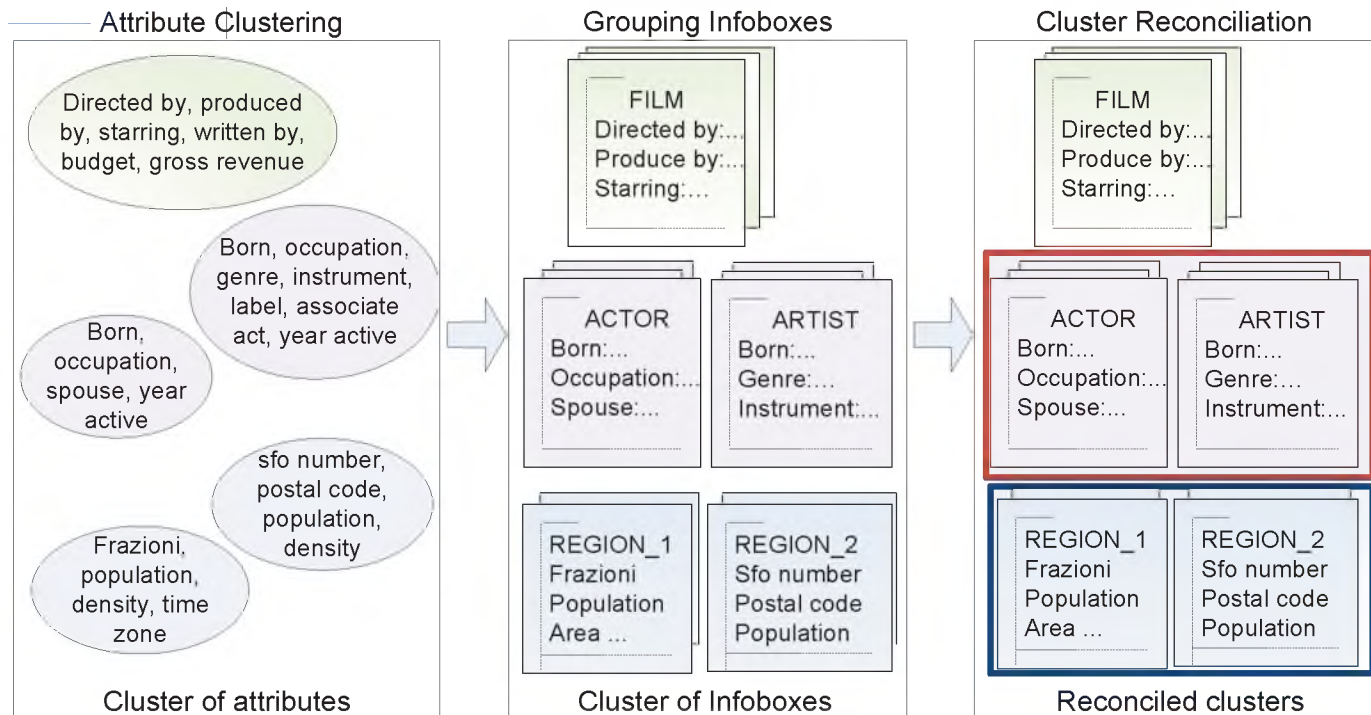


Figure 5.4: Illustration of the three-step entity discovery approach

ib is assigned to an entity type T , whose attribute set is the most similar to its schema. In addition to deriving a set of entity clusters, we also discover relationships that are implied by the links between entities belonging to these clusters. This information is used to build a *schema graph*, where entity types are represented as nodes and the relationships correspond to the link predicates associated with the underlying hyperlinks. The schema graph is used in the *Cluster Reconciliation* step, which merges entity types which have similar relationships in the schema graph. For example, different types of Regions, such as Region_Italy and Region_Swiss, or different types of Person, such as Actor and Artist, are merged.

We then use the discovered entities and relationships to refine the entities. We leverage the relationships between entities to provide finer-grained concepts. For example, entities belonging to type Actor can be refined into subclasses such as Star Actor, Director, Producer, Cinematographer.³ However, the use of different terms to represent the same attribute concept can result in class *fragmentation* where entities of the same type are split into different clusters. To avoid duplicated relationships and help attenuate fragmentation, we perform *schema-matching* across entity types to identify synonym attributes.

5.3 Clustering Wikipedia Infoboxes (WIClust)

A possible solution to cluster infoboxes would be to compute the similarity between all pairs of infoboxes. However, this solution is unlikely to be effective for a number of reasons, including high variability in how infobox attributes are represented; large number of optional and ambiguous attributes; and a skewed distribution of entities. Besides, due to the large number of infoboxes, this approach is expensive. With WIClust, we employ a two-pronged approach to address these limitations: (i) we take advantage of the large number of available infobox schemata and use *attribute correlation* as a source of similarity (and dissimilarity) information; and (ii) before clustering the infoboxes, we discover sets of attributes that are highly correlated and yet distinct from each other and thus are good candidates for describing an entity. An added benefit of WIClust is that it does not require the number of desired clusters to be known a priori: this is obtained naturally from the set of correlated attributes.

5.3.1 Attribute Clustering

The goal of this step is to find, for each entity type, a group of representative attributes. Based on the observation that the important attributes for an entity type T co-occur often in the schema of instances (i.e., infoboxes) of T , our algorithm uses *correlation* to group these attributes together.

³In Wikipedia, there are no specific infobox templates for these entities—they all use the Actor template.

Given two attributes a_p and a_q , there are two types of *correlation* between them: negative (*NC*) and positive correlation (*PC*) [93]:

$$NC(a_p, a_q) = \begin{cases} 0 & \text{if } a_p, a_q \subset ib_k \\ \frac{(C_p - C_{pq})(C_q - C_{pq})}{(C_p + C_q)} & \text{otherwise} \end{cases} \quad (5.1)$$

$$PC(a_p, a_q) = \frac{C_{pq}}{\min(C_p, C_q)} \quad (5.2)$$

where C_p , C_q , and C_{pq} correspond to the number of infoboxes that contain attribute a_p , a_q , and both of them, respectively. These correlations indicate the strength and the direction of a linear relationship between two attributes (e.g., negative—push away, or positive—pull together). Intuitively, *NC* is high when a_p , a_q rarely co-occur in the same schema (C_{pq} is small compared to C_p and C_q). In contrast, *PC* is high when a_p , a_q often co-occur in the same schema (C_{pq} is close to C_p and C_q). The formulas for the correlations are similar with Equation 2.7 and 2.8 in Chapter 2. However, the intuition and the usage of these correlations are different. Attributes that have a high positive correlation are likely to belong to the same entity type. For example, $PC(\text{Directed by}, \text{Starring})$ is high because *Directed by* and *Starring* usually co-occur in infoboxes of the type *Movie*. In contrast, $NC(\text{Directed by}, \text{Occupation})$ is high since these two attributes rarely co-occur in the same schema: *Occupation* appears in the schema of a person while *Directed by* appears in *Movie* schemata. Because there are many kinds of correlation measures [83], it would be interesting to investigate other measures that have the same intuition.

Considering the correlation coefficients among all pairs of attributes, the decision of clustering attributes is based on the following intuition: pairs of attributes with high *NC* scores should be assigned to different clusters, and all pairs of attributes within each cluster should have high *PC* scores. We introduce three constraints ($C1, C2, C3$), shown in Table 5.2, to capture this intuition. Our algorithm then iteratively builds new attribute clusters using previously established ones. Given a set of n established attribute clusters at the current iteration $\ddot{A} = \{A_i | i = 1..n\}$, constraint $C1$ determines whether an attribute a has high *NC* scores (i.e., greater than a threshold T_s) with every attribute a_{ij} of every cluster A_i . If so, attribute a should not be grouped with any previously established clusters. Instead it should be *separated* into to a new cluster. Constraint $C2$ determines whether the *PC* scores between an attribute a and every attribute a_{ij} of each cluster A_i is greater than a threshold T_g . In that case, a should be added to cluster A_i . Finally, constraint $C3$ determines if an attribute a can be merged into a cluster $A_k \in \ddot{A}$ by checking if the grouping constraint $C1$ between a and A_k and the separating constraint $C2$ between a and each remaining cluster $A_i \in \ddot{A}, i \neq k$ are both satisfied.

Although the constraints $C1$ to $C3$ reflect our intuition, they have a limitation. According to the merging constraint, for a given attribute, the grouping constraints must be satisfied by every attribute

Table 5.2: Constraints used in the Attribute Clustering step

#	Action	Constraints
C1	<i>Separate</i> $S(a, \bar{A})$	$NC(a_{ij}, a) > T_s, \forall a_{ij} \in A_i$
C2	<i>Group</i> $G(a, A_i)$	$PC(a_{ij}, a) > T_g, \forall a_{ij} \in A_i$
C3	<i>Merge</i> $M(a, \bar{A}, k)$	$S(a, \bar{A} \setminus A_k)$ and $G(a, A_k), \forall a_{ij} \in A_{i(i \neq k)}$
C1'	<i>SeparateExt</i> $SE(a, \bar{A}, D_s)$	$NC(a_{ij'}, a) > T_s, \forall a_{ij'} \in A_i$ and $ \{a_{ij'}\} / A_i > D_s$
C2'	<i>GroupExt</i> $GE(a, A_i, D_g)$	$PC(a_{ij'}, a) > T_g \forall \{a_{ij'}\} \in A_i$ and $ \{a_{ij'}\} / A_i > D_g$
C3'	<i>MergeExt</i> $ME(a, \bar{A}, k)$	$SE(a, \bar{A} \setminus A_k, D_s), GE(a, A_k, D_g), \forall a_{ij} \in A_{i, i \neq k}$

in a cluster and the separation constraints must be satisfied by all the clusters. As a result, attributes that co-occur with many attributes in a cluster, but not with *all* of them, could be assigned to a new cluster; and infrequent attributes or attributes of low frequent entity types which have low NC scores with all other attributes, could be missing from the cluster. The following example illustrates this limitation. Let `{born, occupation, spouse, domestic partner, website}` be the candidate attributes for the type *Actor*. Applying strict constraints C1 and C2 over these attributes will result in the cluster `{born, occupation, spouse}` because they all present many co-occurrences with each other. Even though `domestic partner` often co-occurs with `born` and `occupation`, it rarely co-occurs with `spouse`, leading to a low positive correlation between them and thus, `domestic partner` will not be grouped into the resulting cluster. To solve this problem, we relax the constraints by requiring only a subset of the attributes within a cluster satisfy the correlation conditions with a given attribute. This relaxation creates three new *soft* constraints C1', C2', and C3', also shown in Table 5.2. We define the relaxation degrees D_s and D_g for these constraints which determine the lower bound for the percentage of attributes within one cluster that must satisfy the separating and grouping conditions, respectively. By applying the soft constraints in the previous example, the derived cluster would contain `{born, occupation, spouse, domestic partner}`.

Another limitation of these constraints is that they do not allow multiple clusters to share common attributes, since according to the separation constraint, the new emerged cluster must have high negative correlation scores with all other clusters. However, this condition is not practical in our data because an attribute may appear in many schemata of different types. We say that such an attribute is *ambiguous*. As a result, the clustering algorithm could either separate the ambiguous attributes across clusters or coalesce the clusters that share the ambiguous attributes. For example, given the schemata of *Movie* and *Show*, respectively, as `{starring, directed by, produced by, written by, website, language, country, runtime}` and `{starring, number of episodes, runtime, country, website, language}`, attributes `website, starring, running time, language, and country` are ambiguous. As a consequence, they will either be split across the two clusters or all attributes of *Movie* and *Show* will be coalesced into a single cluster if we relax the constraints. To avoid this problem,

we identify and isolate potentially ambiguous attributes before running the Attribute Clustering algorithm, and restore these attributes after the entity clusters are derived. Let $G_A=(V_A, E_A)$ be an *attribute graph* where vertices are the attributes, and there is an edge between two vertices if they often co-occur in the same infoboxes (i.e., more than T times). Since an ambiguous attribute belongs to the infobox schemata of multiple entity types, its vertex will have higher centrality than the others. We use *betweenness* [22], a measure of vertex centrality, to distinguish attributes regarding their ambiguity. The betweenness $B(v)$ for vertex v is defined as:

$$B(v) = \sum_{s \neq v \neq t \in V, s \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \quad (5.3)$$

where σ_{st} is the number of shortest paths from s to t , $\sigma_{st}(v)$ is the number of shortest paths from s to t that pass through a vertex v . The intuition of betweenness is that vertices that occur on many shortest paths between other vertices have higher betweenness than those that do not. Therefore, ambiguous attributes which are shared by different entity types will have higher betweenness. We then isolate them by choosing the top K attributes with the highest betweenness. Although there are other centrality measures, e.g., *in-degree*, unlike betweenness, these cannot simultaneously distinguish ambiguous attributes within large and small schemata.

Softening of constraints is still a valid solution because we want to obtain a balance for the attribute precision and recall. Although it can result in the potential loss in attribute precision, high attribute recall is also important for the Infobox Grouping step since this step is based on maximum overlapping/similarity.

We propose the Attribute Clustering algorithm (Algorithm 6) which uses the soft constraints to generate the attribute clusters. Let P be the set of unambiguous attribute pairs present in the infoboxes of all entity types; we iteratively choose from P the attribute pair with the highest negative correlation, integrate them to the appropriate cluster in \vec{A} (line 12), or let them emerge as a new cluster (line 10). We use the soft constraints to deal with optional attributes and use betweenness to isolate ambiguous attributes. Although softening of constraints can result in the potential loss in attribute precision, high attribute recall is also important for the Infobox Grouping step since this step is based on maximum overlapping/similarity. Softening of constraints is still a valid solution because we want to obtain a balance for the attribute precision and recall.

5.3.2 Infobox Grouping

The Attribute Clustering step produces a set of attribute clusters $\vec{A} = \{A_i | i = 1..n\}$, where A_i corresponds to the schema for an entity type T_i .⁴ These are given as input to the Infobox Grouping

⁴Different approaches are possible for assigning labels for types, e.g., labels can be manually selected or set to the most common terms in the category names or template names in the cluster.

Algorithm 6 Attribute Clustering

```

1: Input: Set of unambiguous attributes  $U = \{a_i\}$ 
2: Output: Set of attribute clusters  $\ddot{A} = \{A_i | i = 1, \dots, n\}$ 
3:  $P = \{(a_p, a_q) | a_p, a_q \in U, p \neq q\}$ ,  $\ddot{A} \leftarrow \{\emptyset\}$ 
4: while  $P \neq \emptyset$ 
5:   Remove  $(a_p, a_q)$  with the highest  $NC(a_p, a_q)$  from  $P$ 
6:   if  $\ddot{A} = \{\emptyset\}$ 
7:      $\ddot{A} = \{\{a_p\}, \{a_q\}\}$ 
8:   else if either  $a_p$  or  $a_q \notin \ddot{A}$  (suppose  $a_p \notin \ddot{A}$ )
9:     if  $SE(a_p, \ddot{A}, D_s)$ 
10:       $\ddot{A} = \{\ddot{A}, \{a_p\}\}$ 
11:     else if  $ME(a_p, \ddot{A}, k)$ 
12:        $A_k = \{A_k, a_p\}$ 
13: end while

```

step which produces a set of entity clusters $\ddot{E} = \{E_i | i = 1..n\}$, where each E_i is the set of infoboxes of entity type T_i . We group infoboxes of the same entity type together by assigning each infobox ib to the attribute cluster that is most similar to the schema of ib .

Let S^{ib} be the schema of infobox ib and S^T be the schema of entity type T , which initially contains the attributes of A_T . A possible approach would be to choose T such that S^{ib} is the most similar to S^T . However, the initial similarity between S^{ib} and A_T may be low because a given attribute cluster may not contain the complete set of attributes; notably, it may be missing both optional and ambiguous attributes. To repair potential omissions in the attribute clustering step and improve the similarity between the infobox and the cluster where it may belong, we apply a bootstrapping approach which gradually updates the set of attribute clusters with attributes from the schema of their own instances. In particular, we assign ib to type T whose attribute cluster A_T has the highest similarity to S^{ib} and that is also above a threshold θ . These infoboxes form the initial set of entity clusters and we use them to build a *probabilistic attribute model* for each entity type T , where each attribute $a_i \in S_T$ is assigned a weight w_i using TFIDF [10]—the intuition here is to give higher weights to important attributes. We then compute the similarity between an unassigned infobox schema S^{ib} and every attribute set S^T using Equation 5.4 and then assign to ib the type T which has the highest similarity.

$$sim(S^{ib}, S^T) = \frac{\sum w(a_i | a_i \in S^{ib} \cap S^T)}{\sum w(a_j | a_j \in S^{ib} \cup S^T)} \quad (5.4)$$

5.3.3 Efficiency

We note that while some of the baseline clustering methods we explore in our experimental evaluation require pairwise comparisons between all infoboxes, the Infobox Grouping step is *linear* with respect to the number of *infoboxes*; and the cost of the Attribute Clustering step is proportional to the number of *attributes*. Given the fact that the number of infoboxes is much bigger than the

number of attributes, our approach ends up being computationally more efficient than these other methods.

5.3.4 Cluster Reconciliation

The goal of Cluster Reconciliation is to reconcile and merge entity clusters that correspond to the same concept. By taking only the infobox schema into account, we may miss entities that are similar and yet have different sets of attributes. For example, the entity types for *French_Region* and *Swiss_Region* are distinct because most of their attributes differ, but they both represent similar geographical concepts. We observe that the entity instances of similar types often have similar relationships to other types. For example, entity instances of the types *Actor* and *Artist* often have the same relationships *Born* with different *Region* types, and the relationship *Starring* with *Movie* (e.g., Figure 5.5). Thus, we leverage the link structure between infoboxes as another source of similarity between entity clusters to merge the ones that are similar. Specifically, we combine the discovered entity clusters and the link predicates that connect them to build a two-layer graph, in which the top layer—the *schema graph*, connects different entity types with their relationships; and the second layer—the *instance graph*, consists of infoboxes for the corresponding entity types with the hyperlinks that connect them. *Schema Graph* $G = (V, E)$ is a labeled directed graph where each node $v_T \in V$ represents an entity type T . There is an edge $e_p(v_T, v_{T'}) \in E$ from node v_T to $v_{T'}$ if there is a link with predicate p from the attribute value of an entity in v_T to an entity in $v_{T'}$. We define the link predicate as the attribute name whose value contains the link.

Given a node v_T in the schema graph, we say that nodes that link to v_T *co-cite* v_T , and nodes that v_T links to are *cited by* v_T . Recall that given two nodes, there can be multiple predicates linking them. We define a structural similarity factor $SSF(v_T, v_{T'})$ as the total number of common nodes (entity types) that co-cite and are co-cited with the same predicate p_k between v_T and $v_{T'}$, for all predicates p_k between v_T and $v_{T'}$. Hence, $SSF(v_T, v_{T'})$ will be higher if v_T and $v_{T'}$ share more of

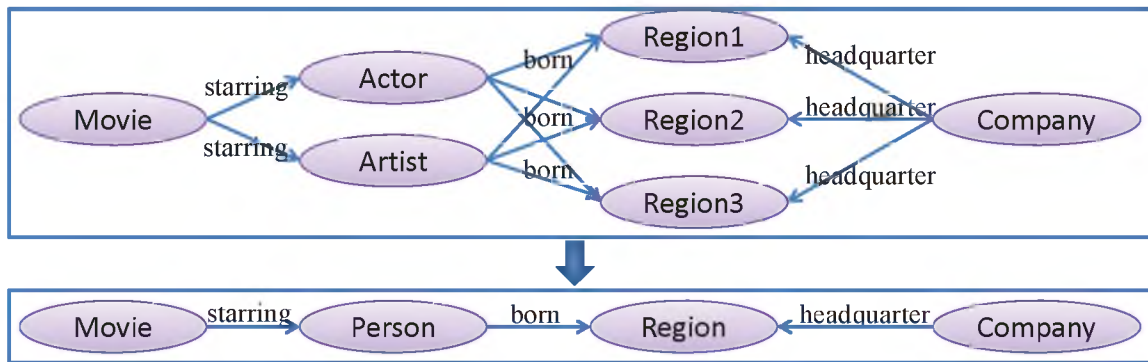


Figure 5.5: Cluster Reconciliation using link structure and schema similarities

the same relationships with other types. The final similarity between entity clusters $E_T, E_{T'}$ is the cosine similarity of their associated probabilistic attribute models (Section 5.3.2) multiplied by the SSF factor. The intuition is to leverage the similarity of relationship structure between types from the schema graph as a factor to boost the syntactic similarity and help merge similar entity types.

WIClust is similar to collective entity resolution [16] which combines the attribute similarity and the relational similarity. There are a few neighborhood similarity measures for collective resolution, such as common neighbors, Jaccard coefficient, Adamic/Adar similarity, Adar similarity with ambiguity estimate, and higher-order neighborhoods [1, 66, 16]. The idea of using co-citations as a measure of similarity has also been used for clustering Web-forms [13], but we are not aware of other papers that do this for infoboxes or schema-matching. Different to collective entity resolution, which is defined at the level of individual instance, WIClust is defined for sets of aggregated instances, e.g., entity types. Our structural similarity considers both entity type and predicate information from the link. We note that, for calculating the SSF factor, it is necessary to use *both* co-citation and link predicates to avoid irrelevant information. Co-citation alone is not sufficient because there are entity types that co-cite or are co-cited by the same set of types, and yet, are not similar. For example, Person and Company both co-cite Region, and both Music and Company are co-cited by Album. However, neither Person and Company nor Music and Company are similar. Likewise, using only *predicates* ignoring node types also leads to problems. For instance, the Movie type links to both (distributed) Company and Country types via the predicate `Distributed by`⁵. Thus, given the heterogeneity and incompleteness of Wikipedia, approaches that leverage only the link predicates [109] will fail because of propagated errors.

5.4 Semantic Refinement and Synonyms Identification

To support and enrich the querying system, it is useful to refine the semantics of the entities and find synonym attributes. Naïvely using infobox schemata may not always be sufficient to classify entities into fine-grained concepts. We observe that, by taking into account the links between entities, it is possible to refine entity types. For example, the relationships between types Person and Movie enables Person to be refined into more specific subclasses, including Actor, Director, Producer, Writer, Cinematographer, Editor, and Composer.

Let v_T and $v_{T'}$ be two adjacent nodes in the schema graph G , and P be the set of predicate links from $v_{T'}$ to v_T . We partition the set E_T of entities with type T into subsets $\{E_k^T\}$ where E_k^T is a subset of entities with type T that are linked from T' by the predicate $p_k \in P$. For example, in Figure 5.6, the initial cluster of entity type Person is partitioned into subclusters Actor, Director,

⁵http://en.wikipedia.org/wiki/The_Street_Fighter

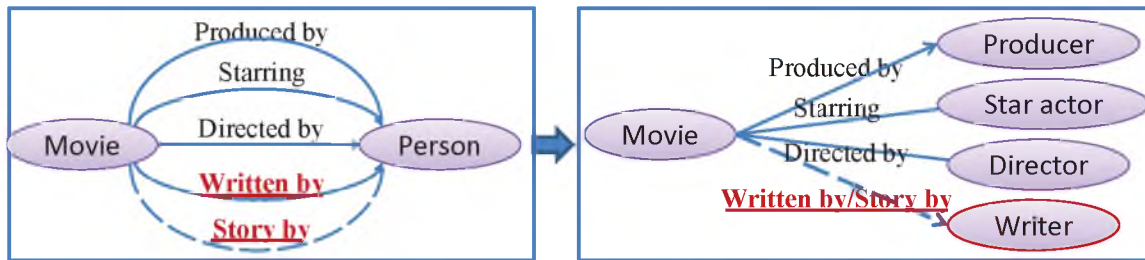


Figure 5.6: Example of Semantic Refinement

Writer, and Producer. However, given the variability in how attributes are represented (e.g., *Born* and *Origin*; *Story by* and *Written by*), naively applying semantic refinements leads to undesirable *fragmentation*: entities of the same type are split into different clusters. Thus, a schema-matching step to find *synonym* attributes is necessary to avoid fragmentation and increase query recall.

Due to the syntactic and semantic heterogeneity present in attribute labels, simple schema-matching techniques that rely just on string similarity are not effective. For example, *Religion* and *Rel* (short for *Relative*) are not synonyms, while *Starring* and *James Bond* are.⁶ Likewise, attribute values are heterogeneous and not uniform across entities i.e., different units, resolutions, and styles are used, and many attributes contain composite values, e.g., *born* contains both birthdate and birthplace information.

To identify synonyms, we can use attribute correlations but with a different goal and intuition from the one used in Section 5.3.1. Here, synonym attributes are semantic alternatives and rarely co-occur in the same schemata—their *NC* is high. For example, since attributes *known for* and *notable works* rarely co-occur in the same infobox, they can potentially be synonyms. Positive correlation, in this case, is only used to group attributes that are matched to the same attribute. For example, *james bond* and *also starring* are grouping attributes in James Bond movies, and both are matched to *starring*. Thus, in order to find synonyms, we need to cluster *negatively correlated* attributes. Although correlation can be used to find synonyms, in the presence of optional and rare attributes, there is insufficient evidence, and consequently, they have artificially high negative correlation scores with other attributes, resulting in propagation of errors [53, 93].

To attenuate errors stemming from optional and rare attributes, we reinforce the negative correlation with other sources of similarity information such as attribute names. We also use a new kind of information that is available in Wikipedia: the *link type*. Because we have the entity schemata, we can infer the link type of an attribute, i.e., the type of the object to which an attribute links.

⁶*James Bond* is an attribute name in James Bond Movies that has similar meaning as *Starring*. Although *James Bond* and *Starring* have different semantics, identifying the correspondences between them is useful for query applications to increase recall.

For example, the link type of `Place of birth` is *Region*, while the link type of `Directed by` is *Person*. We then combine all this information, i.e., correlation, attribute names, and link type, using a prudent matcher [81]. By iteratively identifying synonyms, we can remove redundant relationships among entity types and avoid type fragmentation (e.g., Figure 5.6). Also, note that using negative correlations to find synonyms only makes sense when considering attributes of the same or from similar entity types. The Entity Clustering process helps define the entity schemata and significantly reduces the candidate space for synonym identification. Since the problem of schema discovery and synonym discovery is a ‘chicken and egg’ problem, similar to collective entity resolution [16], integrating them together in an iterative process could be a future direction for our work.

5.5 Experimental Evaluation

To evaluate the effectiveness of our approach to entity clustering, we compare it against other clustering techniques in terms of the precision, recall, and cohesion of the derived clusters. We also compare the types automatically generated by our approach against the types in DBpedia. In addition, we assess the precision and coverage of the derived attribute clusters as well as study the sensitivity of `WIClust` to different threshold values. Finally, we present an evaluation of the structure reconciliation and semantic refinement.

We used correlation thresholds $T_s=5$ and $T_g=0.4$ for all three domains. We discuss the effects of choosing different correlation thresholds in Section 5.5.4. We ran our experiments on a PC with an Intel 2.67GHz, 8GB Memory, running Windows 2003. It took less than 10 minutes to cluster all the infoboxes in the Movie domain, which is the largest.

5.5.1 Dataset and Evaluation Metrics

5.5.1.1 Dataset

We extracted 107,000 infoboxes (HTML tables) from Wikipedia pages in three different domains: Movie, Book, and Computer. Because we extract the HTML contents, we can obtain infoboxes that use implicit or generic templates. Table 5.3 shows some statistics for these domains.

To evaluate precision and recall of our clusters, we created the gold data (e.g., the correct data) by randomly selecting 400 infoboxes from each domain and manually labeling the selected infoboxes with a class name. The choice of classes was guided by our goal of obtaining an intuitive query interface. For example, in the `Movie` domain, we considered classes such as `Movie`, `Show`, `Album`, `Artist`, `Actor`, `Writer`, `Country`, `State`, `City`, etc. We should note, however, that this choice is subjective: there are different ways of classifying the infoboxes. For example, one can use fine-grained classes (e.g., `Eurovision Song Contest Entry`, in addition to `Song`) or coarse-grained

Table 5.3: Description and characteristics of the dataset

Domain	Movie	Book	Computer
#infoboxes	48K	31K	28K
#attributes (#unique atbs)	371K (7986)	251K (6019)	249K (4889)
#values (#unique values)	403K (214K)	263K(133K)	262K(127K)
#categories (#unique categories)	599K (68K)	438K (69K)	221K (32K)
#entity types in the gold data	57	57	58
% Implicit Infobox	5.70%	10.10%	5.60%
% Incorrect assignments in DBpedia	7.00%	6.20%	5.90%
% Missing instances in DBpedia	12.70%	22.10%	25.60%

ones (e.g., Person instead of Artist and Actor). Because this choice can influence the results we report below, we have also associated with each infobox the label provided by a third independent party; specifically, for each infobox, we checked if it was present in DBpedia and if so, we assigned to it the class label provided by DBpedia. We note that, since the class assignment in DBpedia may contain mistakes, we have manually checked each label and corrected it when necessary. While DBpedia assigns each Wikipedia page to an entity type, some pages may contain more than one entity. For example, in some pages for movies, there are two infoboxes, one for the movie and one for its soundtrack; DBpedia would assign both infoboxes to the same class. To evaluate the cluster reconciliation algorithm (Section 5.5.3), guided by the DBpedia ontology, we grouped together similar entity types (e.g., Actor, Artist, and Writer into Person).

An overview of our gold data is given in the second half of Table 5.3. It shows the percentage of the instances in our dataset which are not present in the latest version of DBpedia, and for the instances that are in DBpedia, the percentage of those that are assigned to incorrect categories. Note that less popular domains, such as Computer, have low coverage by DBpedia and so 25.6% of instances that appear in our sample do not appear in DBpedia.

5.5.1.2 Evaluation Metrics

To evaluate the entity clusters \vec{E} , we use the F-measure. Given an entity type T , let E_T be the entity cluster discovered by our algorithm and E'_T be the correct entity cluster of T . The precision and recall for type T are defined as follows:

$$Pr(E_T) = \frac{|E_T \cap E'_T|}{|E_T|}, \quad Rc(E_T) = \frac{|E_T \cap E'_T|}{|E'_T|} \quad (5.5)$$

We compute the weighted average of precision and recall according to the number of elements for each entity cluster [50]. F-measure is the harmonic mean of precision and recall.

We use *cohesion* to assess how homogeneous the derived clusters are. We compute cohesion by averaging the distance between any two elements inside a cluster:

$$Cohesion(E_T) = \frac{\sum_{x,y \in E_T} proximity(x,y)}{number\ of\ element\ pairs\ (x,y)} \quad (5.6)$$

To evaluate the quality of the attribute clusters generated by the Attribute Clustering algorithm, we compute attribute precision and coverage:

$$Atb_Pr(A_T) = \frac{\sum w(a_i | a_i \in A_T \cap A'_T)}{\sum w(a_i | a_i \in A_T)} \quad (5.7)$$

$$Atb_Cv(A_T) = \frac{\sum w(a_i | a_i \in A_T \cap A'_T)}{\sum w(a_i | a_i \in A'_T)} \quad (5.8)$$

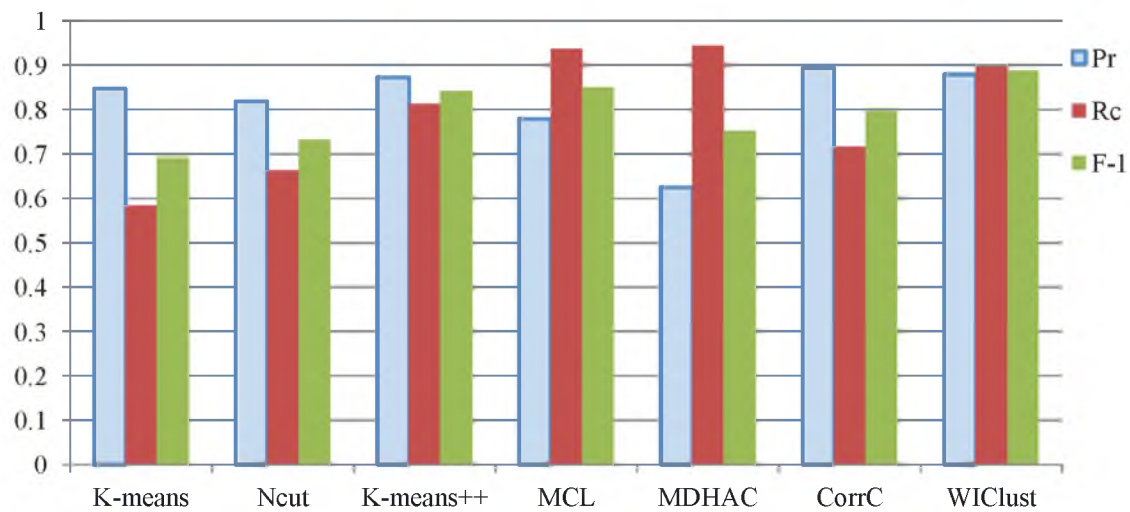
where A_T is the attribute cluster derived by our algorithm for an entity type T , A'_T is the union of all attributes in the entity cluster for T in the correct data, and the attribute weight $w(a_i)$ is the frequency of a_i in type T in the attribute cluster (Equation 5.7) and in the correct data (Equation 5.8). Attribute precision reflects the number of attributes correctly identified by `WIClust`, while attribute coverage measures the recall of the attribute set that we found over the actual set of attributes for an entity type. We report the average precision and coverage of the attribute clusters according to the number of elements for each entity type.

5.5.2 Quality of the Entity Clusters

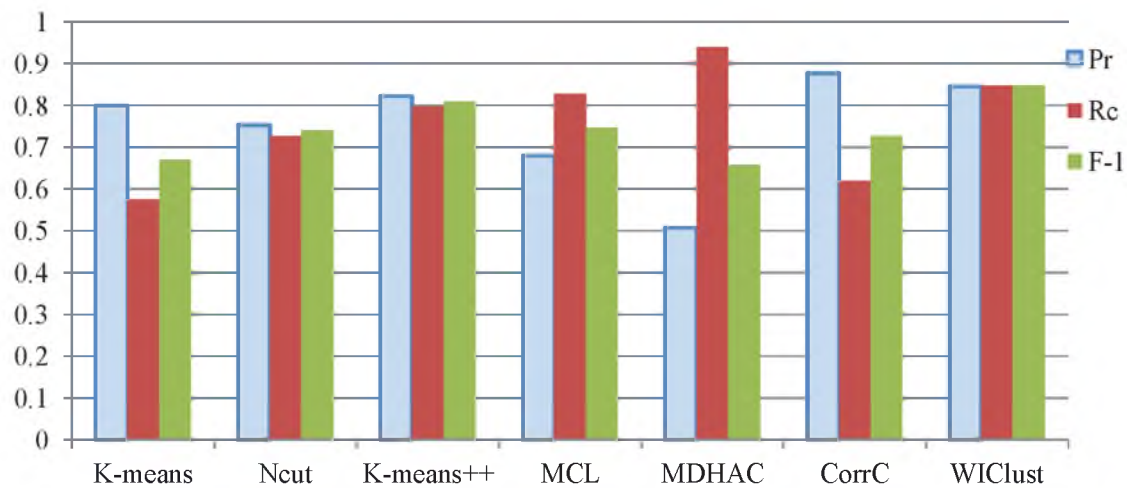
5.5.2.1 Comparing against Different Clustering Methods

We compare the results of our clustering algorithm against a series of state-of-the-art clustering methods: K-means, K-means++ [6], Normalized Cut (NCut) [90], and Markov Clustering (MCL) [50, 98]. We also present a comparison against: MDHAC, a new categorical clustering algorithm which has been shown to outperform many categorical clustering algorithms like ROCK, COOLCAT, and CATUS that have been used to cluster Web-form schemata [54]; and Correlation Clustering (CorrC) [4], which is similar to `WIClust` and relies on correlation. We use Jaccard [10] as the similarity measure for all methods, except for MDHAC, which uses the chi-squared test and CorrC, which uses the attribute correlation. We use the number of clusters in the correct data as the target number of clusters for these clustering methods. We should note that *WIClust does not require the number of clusters and it still outperforms the other clustering techniques*. The results are shown in Figure 5.7.

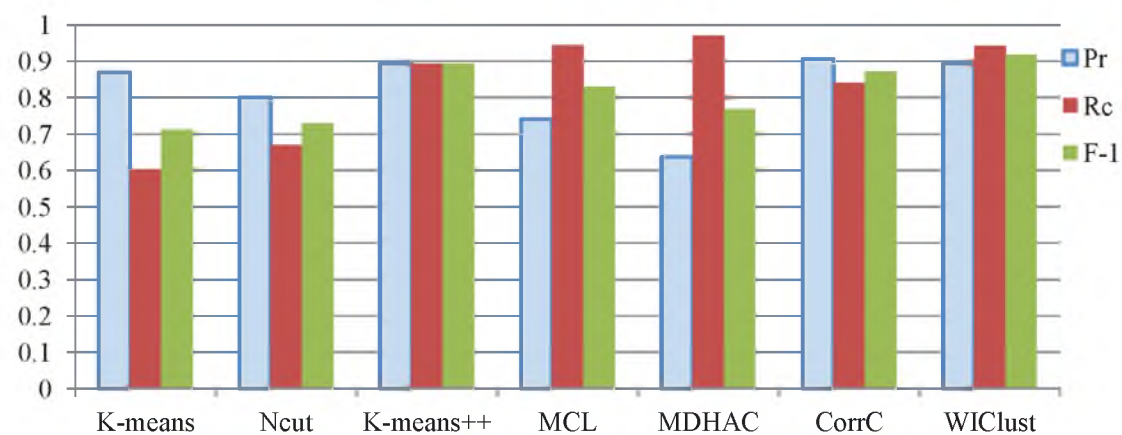
Our approach obtains both high precision and recall: the F-measure values for `WIClust` are the highest for all domains—the marginal gains vary from 4% to 20% in Movie, 4% to 19% in Book, and 2% to 20% in Computer domain. To ensure that our findings are significant, we computed the



(a) Movie



(b) Book



(c) Computer

Figure 5.7: Result of the entity clusters by using different clustering strategies in Movie, Book, and Computer domain

Wilson interval with the level of significance $\alpha = 5\%$ [23]. Table 5.4 shows the confidence interval of `WIClust` in the three domains. We can see that the error margin is small in all cases.

K-means has the lowest performance, likely because it heavily depends on the choice of initial seeds. NCut maximizes the group association inside each cluster and minimizes the disassociation between different clusters, which leads to high precision but low recall because it tends to split big clusters into smaller clusters with similar sizes. K-means++ outperforms K-means and NCut. This can be attributed to the fact that it chooses the furthest points as initial seeds. However, because there is a large variation in the sizes of the clusters, some of the initial points fall into the same cluster. This results in fragmentation and a lower recall.

MCL simulates stochastic flows by performing random walks on a similarity graph based on the intuition that the random walks strengthen the flow where the similarity connections are dense and make the underlying cluster structure become visible through iterations.⁷ However, it tends to merge small clusters together, resulting in high recall but at the cost of lower precision. MDHAC assumes that homogeneous sources share the same generative attribute model. It uses the χ -square hypothesis test to maximize the statistical heterogeneity among clusters through the clustering process. However, in this scenario, it tends to merge together unrelated schemata which contain ambiguous attributes. These mistakes are propagated through the iterative clustering process, leading to a low precision.

CorrC pulls all the correlated neighbors for each random pivot. Thus, the grouping condition is loose and the algorithm is sensitive to the grouping threshold. To attenuate this problem, we used the same mechanism and thresholds that we applied in `WIClust`: enforce stricter requirements (e.g., require all attributes within a cluster to satisfy the grouping conditions) and soft constraints (e.g., require only a subset of the attributes within a cluster to satisfy the condition). However, since CorrC depends on the order in which the pivots are selected, if weak pivots are selected early (e.g., ambiguous or optional attributes), fragmented attribute clusters are derived because the correlation between these attributes and others is not strong, which results in a lower recall for all domains.

⁷We varied the inflation parameter in MCL to tune the coarseness and quality of the clustering.

Table 5.4: Accuracy and the confidence interval of `WIClust`

Domain	Movie	Book	Computer
Sample size	401	417	375
Confidence interval	3.1%	3.4%	2.7%
WIClust F1	89%	85%	92%

5.5.2.2 Comparing with DBpedia

Besides our correct data, we also used the (corrected) DBpedia types to compute the F-measures for WIClust on the overlapping instances between our sample and DBpedia. The results are similar to the ones reported in Figure 5.7. The F-measure for the Movie domain is 87.4%, for Book 86.0%, and 93.1% for Computer. Compared to the values shown in Figure 5.7, the values for Book and Computer are slightly higher. This is due to the fact that, for these domains, the infoboxes in our dataset which overlap with DBpedia are more homogeneous.

To validate our approach, we also computed *cluster cohesion* (Equation 5.6) using proximity based on the DBpedia ontology. We reconstructed the hierarchy of DBpedia ontology [28] and considered only entities in the sample set that have a corresponding DBpedia type. The proximity between two elements x and y in a cluster is calculated as the topological distance between these two objects (i.e., the number of edges) and their lowest common ancestor (LCA) in the DBpedia ontology. The closer the nodes are in the tree, the smaller their distance. Higher penalties are given if the LCA is above the boundary nodes, i.e., nodes which distinguish high-level concepts like *Person* and *Organization*. For example:

$LCA(Actor, Movie) = Thing$	$proximity(Actor, Movie) = 10$
$LCA(Actor, Politician) = Person$	$proximity(Actor, Politician) = 3$
	$distance(Person, Politician) = 1$
	$distance(Person, Actor) = 2$

As shown in Figure 5.8, in all domains, the cohesion of WIClust clusters is the lowest (0.32 in the Movie and Computer domain, 0.45 in the Book domain), which means WIClust clusters are

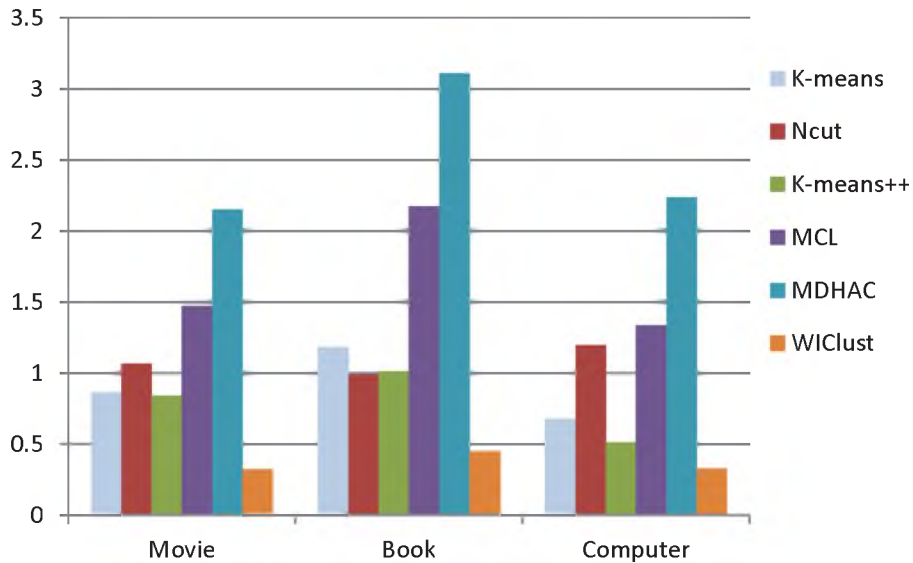


Figure 5.8: Cluster cohesion based on DBpedia ontology

more homogeneous and are very close to the DBpedia ontology. Cohesion for K-Means, NCut, MCL, and MDHAC are higher than K-means++ and WIClust. This is consistent with what we have observed in Figure 5.7. A closer look at the derived clusters shows that cohesion is high for MCL and MDHAC in the Book domain because they cannot separate entity types for Writer, Artist, and Actor.

5.5.2.3 Covering New Templates and Discovering New Entity Types

Because of schema drift and proliferation, there can be many template names associated to an entity type. For example, WIClust discovered templates Infobox Movie, Bond film, Japanese film, Chinese film, and Korean film appearing together with Infobox Film for the entity *Film*. Note that some of these templates are not covered by current mapping rules in DBpedia [8]; for instance, James Bond films are not recognized as Film in DBpedia. For less popular domains like Computer, there are many types that are not covered by DBpedia ontology. Some of the entity types discovered by WIClust which are not covered by DBpedia include: Computer, Processor, Connector, Socket, Calculator, FileSystem, FileFormat, Font, ProgrammingLanguage, SoftwareComponent, OSFamily, GameReview, Animanga, Comics, etc. Therefore, using our approach can help to suggest more templates and improve the coverage of manual approaches like DBpedia.

5.5.3 Reconciliation and Semantic Refinement

5.5.3.1 Reconciled Entity Clusters

The Structure Reconciliation step is applied to merge semantically similar entities which have low syntactic similarity. Table 5.5 shows the precision and recall of the clusters before and after the merging. Before merging, recall is lower because similar classes are still separated and precision is slightly higher because our clusters already merge entity types that have similar schemata (e.g., Album, Song, and Single). After merging, recall increases 4% in the Movie domain and 17% in the Book domain as a result of combining, for example, *Movie*, *Show*, and *Episode*. For the Computer domain, recall increases only marginally since although we can merge a few types like *OS* and *OS Family*, *Video Game* and *Game Series*, we still fail to merge *Video Game* and *Software*, whose attribute sets are different but they appear frequently in the domain. In this case, retrieving a more complete schema graph could help strengthen their similarity. After merging, precision slightly decreases because of some imperfect merges like combining Political Party and Company.

Table 5.5: Entity clusters before and after reconciliation

Domain	Movie		Book		Computer	
	before	after	before	after	before	after
Pr	0.941	0.943	0.960	0.956	0.918	0.914
Rc	0.733	0.776	0.676	0.842	0.771	0.773
F1	0.824	0.851	0.794	0.895	0.838	0.838

5.5.3.2 Synonyms and Semantic Refinement

Table 5.6 shows some of the synonym attributes that we found for entity types *Film* and *Person*. Note that identifying synonyms like *Written by*, *Story by*, and *Novel by* helps us avoid the problem of class fragmentation.

By exploiting links between entities that have complex semantic relationships like *Film* and *Person*, we can identify subclasses, including, for example, *Actor*, *Director*, *Producer*, and *Writer*. We evaluate the accuracy of the refined subclasses by taking a sample of 100 instances of *Actor* and manually checking each of their refined roles and their assigned categories. From these 100 instances, only 22 were not assigned correctly or missed at least one role in their Wikipedia categories. This indicates that our approach can be helpful in improving the quality of Wikipedia data. We note that such refined entity types are not available in DBpedia or derived by template-based approaches [105].

In addition, we should note that we discovered a large number of relationships: 562 in *Movie*, 453 in *Book*, and 446 in *Computer* domain. These numbers are much larger than what was obtained by YAGO, which originally identified only 15 relationships which were then extended to 99 relationships [95].

5.5.4 Attribute Clusters

Even though our main goal is to discover the types for each entity, we also want to investigate the effectiveness of *WIClust* at identifying the schema for each type. In particular, we are interested in the coverage and precision of the attribute clusters resulting from the Attribute Clustering step. Table 5.7 shows the coverage and precision of the discovered attribute clusters in the *Movie* domain.

Table 5.6: Some synonym attributes identified

Entity Type	Synonyms
Film	Starring ~ (James Bond, also star); Written by ~ novel by ~ story by
Person	Nation ~ origin ~ birthplace; alma mater ~ train; known for ~ notable work ~ work ~ notable work role; notable award ~ award; house ~ royal house; literary movement ~ movement; bury ~ burial

Table 5.7: Precision and coverage of the attribute clusters in the Movie domain

Entity type	Precision	Recall	F1
LANG	1.00	1.00	1.00
MUSIC	1.00	1.00	1.00
LANG_FAMILY	1.00	1.00	1.00
COUNTY	1.00	1.00	1.00
CURRENCY	1.00	1.00	1.00
HERITAGE	1.00	1.00	1.00
FILM	0.97	1.00	0.99
BOXER	0.94	1.00	0.97
UNIVERSITY	0.93	1.00	0.96
YEAR	0.88	1.00	0.93
ARTIST	0.85	1.00	0.92
ACTOR	0.85	1.00	0.92
DNS	0.91	0.91	0.91
FICTIONAL_CHAR	0.89	0.91	0.90
ALBUM	0.81	1.00	0.89
COMICS_CHAR	0.79	1.00	0.88
EPISOD	0.78	1.00	0.88
BOOKS	0.83	0.90	0.87
RECORDLABEL	0.75	1.00	0.86
MILITARYCONFLICT	0.75	1.00	0.86
POLITICAL_PARTY	0.75	1.00	0.86
PLAY	0.75	1.00	0.86
COMPANY	0.88	0.83	0.85
TOYCHAR	0.80	0.80	0.80
ADULT_ACTOR	0.67	1.00	0.80
SPORTSEASON	0.67	1.00	0.80
AREA&STATE	0.69	0.91	0.78
WRITER	0.63	1.00	0.77
SINGLE (SONG)	0.60	1.00	0.75
COMEDIA	1.00	0.60	0.75
MoP	0.73	0.75	0.74
OFFICEHOLDER	0.64	0.88	0.74
COUNTRY	0.71	0.75	0.73
SOLDIER	0.83	0.63	0.71
SHOW	0.67	0.71	0.69
DEVICE	0.50	1.00	0.67
TOUR	1.00	0.50	0.67
GOVERNER	0.71	0.63	0.67
TVSTATION	0.53	0.88	0.66
CITY	0.51	0.84	0.64
PRIMEMINISTER	0.80	0.50	0.62
FOOTBALLER	0.40	1.00	0.57
MODEL	0.57	0.50	0.53

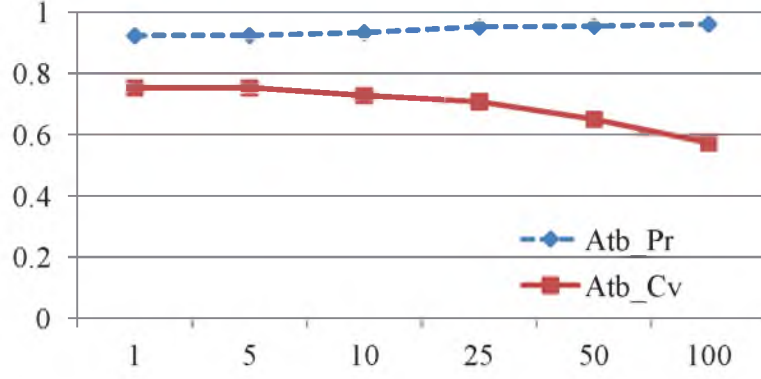
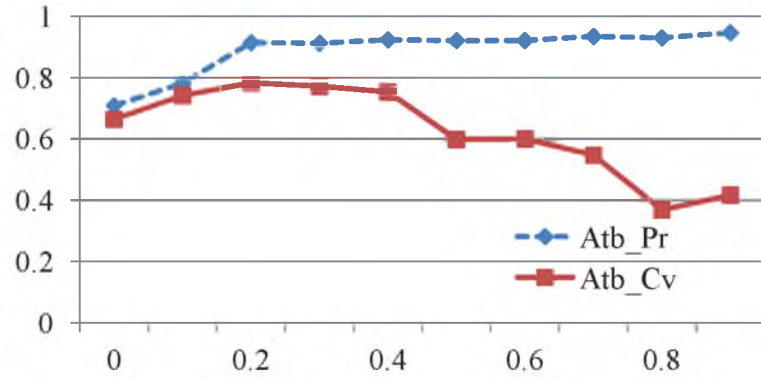
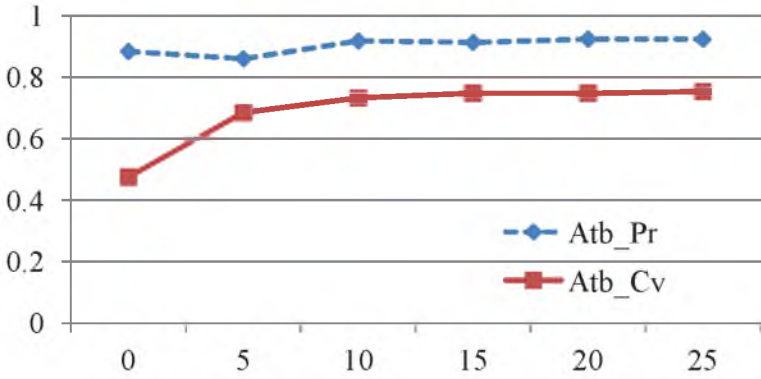
Most of the attribute clusters have very high precision and also high coverage, which means they contain representative attributes for these entities. For example, the attribute clusters for *Film*, *Actor*, *Artist*, *Writer*, *Show*, and *Album* have precision close to 1 and coverage varying from 0.72 to 0.96. Types that have lower scores are either heterogeneous, e.g., *City*, or infrequent and contain many optional and ambiguous attributes, e.g., *PrimeMinister*, *Footballer*.

Figure 5.9 shows the precision and coverage of the attribute clusters when varying the values for the configuration parameters. To study the sensitivity of our approach, we vary one parameter at a time and keep the remaining attributes unchanged. In Figure 5.9(a) and 5.9(b), precision increases but coverage decreases more significantly as T_s and T_g increase. Higher T_s and T_g values mean higher correlation thresholds, i.e., stronger separation and grouping conditions, resulting in higher precision and lower coverage for the attribute clusters. The reduction in coverage is more significant for T_g since it is crucial for grouping the attributes. Figure 5.9(c) shows the effectiveness of isolating top-k ambiguous attributes (e.g., attributes with highest betweenness), which achieves a significant increase in attribute coverage.

Figure 5.10 shows the attribute precision and coverage when changing the degrees of separation (D_s) and grouping (D_g). We can see from this figure that the constraint relaxation is effective, and that it provides a trade-off between precision and coverage. In Figure 5.10(a), we see that precision decreases when D_s and D_g decrease (e.g., more relaxation for separating and grouping constraints). We also observe that if D_g is too small, precision is low because irrelevant attributes are grouped, which misleads the clustering algorithm. Figure 5.10(b) shows that the relaxation for separating and grouping constraints leads to higher coverage. However, too much relaxation (e.g., small D_s and D_g) also makes the precision, and thus the coverage, decrease with the same explanation as above.

5.6 WikiQuery: A Case Study

The entity types and relationships discovered by `WIClust` are useful for systems that support structured queries over Wikipedia content [61, 78]. To assess the benefit of the clusters identified by `WIClust`, we have used them in conjunction with WikiQuery [78]. WikiQuery provides a simple query interface where users can formulate queries using a conjunction of constraints. WikiQuery returns a series multidocument answers. The query *Find films directed by James Cameron that grossed over 100 million dollars and actors born in England*, can be expressed as follows: $Q: \text{Film}(\text{gross revenue} > \$100 \text{ million}) \text{ and } \text{Director}(\text{name} = \text{James Cameron}) \text{ and } \text{Actor}(\text{born in} = \text{England})$. To answer this query, WikiQuery searches the Wikipedia graph induced by the infoboxes (including their types, relationships, and attribute), and constructs answers that consist of Steiner trees [60] in the *instance graph*, where the leaf nodes respect the constraints specified in the query. We use the entity types and relationships returned by `WIClust` to build the Schema Graph in

(a) T_s (b) T_g 

(c) Number of ambiguous attributes

Figure 5.9: Precision and Coverage of the attribute clusters when changing T_s , T_g , and the number of ambiguous attributes

Figure 5.11 which summaries the relationships among different entities. This graph can guide user's queries and the knowledge exploration process.

To assess the usefulness of the entity types discovered by *WIClust*, we compare WikiQuery using *WIClust* types against *Wikipedia* categories. Note that there is a large number of categories in Wikipedia (over 68K categories in Movie domain). Thus, we chose the top-1000 most frequent categories and ran WikiQuery on the same set of queries. We ran experiments over a set of 12

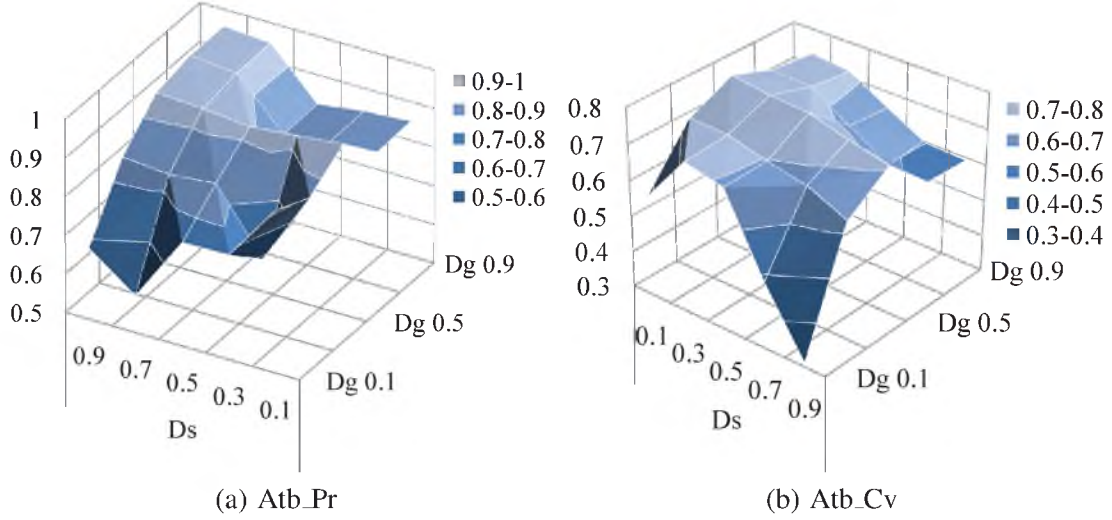


Figure 5.10: Precision and coverage of attribute clusters for different values of D_s and D_g

structured queries on Movie domain. For each query, we selected the top 20 results and presented them to five evaluators who labeled each result as "highly relevant", "relevant", "somewhat relevant", "unclear", or "not relevant". We used the normalized discounted cumulative gain (NDCG) to evaluate the results. NDCG has been widely used in IR benchmarking [57] as well as in other search and ranking problems [62].

Figure 5.12 shows that the NDCG score using entity types discovered by WIClust is always higher. There are many reasons for these better results. First, the categories in Wikipedia are folksonomy-oriented, heterogeneous, and ambiguous. Consider, for example, the case where an entity Actor is assigned to a category "Film Actors", which can be interpreted as the entity being associated with the two concepts: "Film" and "Actor". Extracting the real meaning of each category in such cases is not trivial and the query engine will invariably retrieve irrelevant answers. Second, Wikipedia categories often give very specific semantics to each entity, e.g., *American Horror Films* and *Irish Directors*. Categories with very specific semantics weaken the relationship between entities. For example, while a *Director* always directs some *Film*, only a few *Irish Directors* direct *American Horror Films*. As a result, systems like WikiQuery which use the importance of relationships to construct and rank answers can be misled. Last, but not least, if only the 1000 most frequent categories are used, meaningful yet not very frequent categories, such as *Book*, *Author*, etc. will not be present, greatly limiting the coverage of the queries.

5.7 Related Work

Similar to WIClust, YAGO [95] and DBpedia [8] also extract entity types in Wikipedia, but they do so in different ways. Assuming that there exist certain design guidelines or a common visible

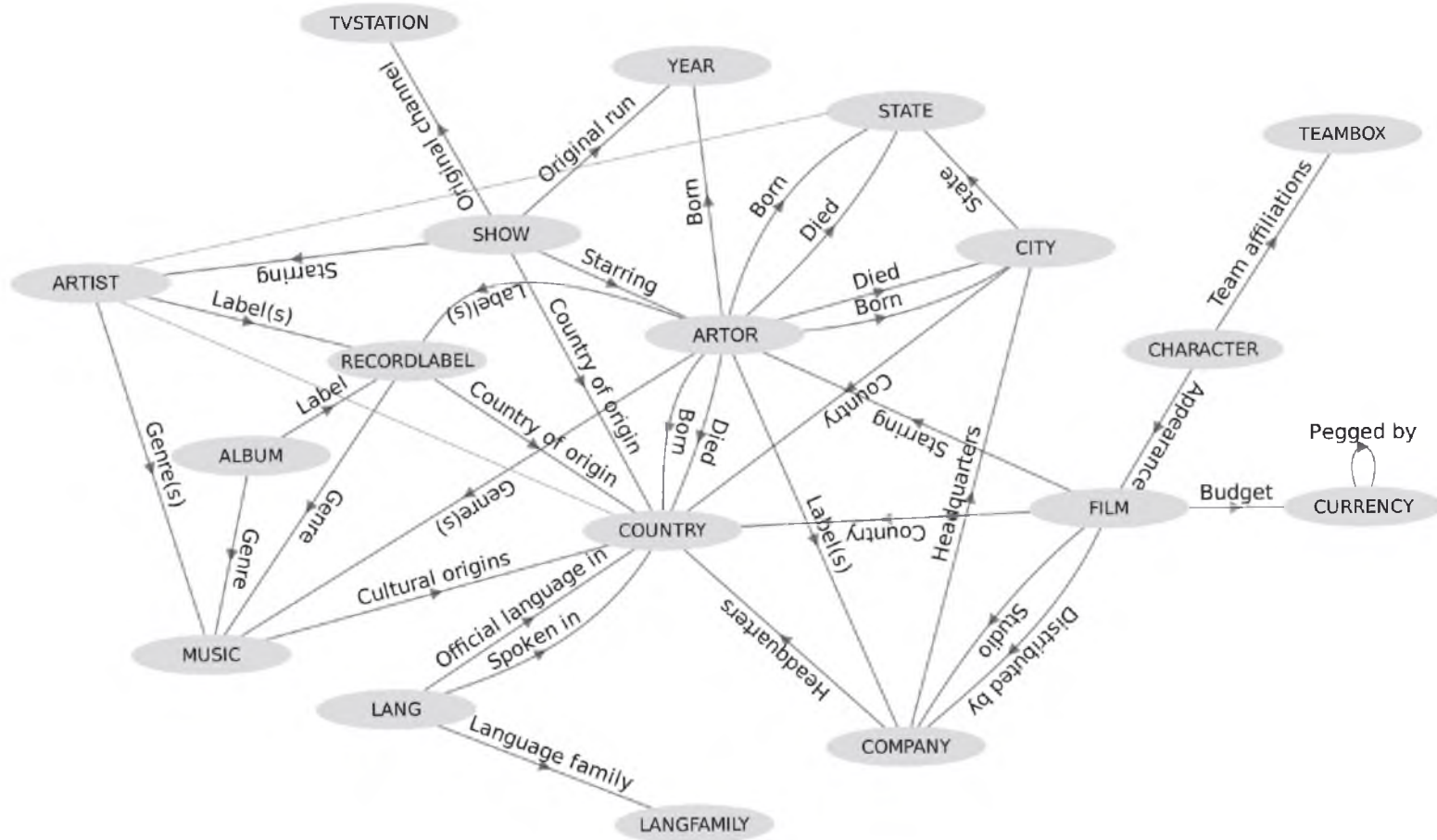


Figure 5.11: Excerpt of the Schema Graph in the Movie domain

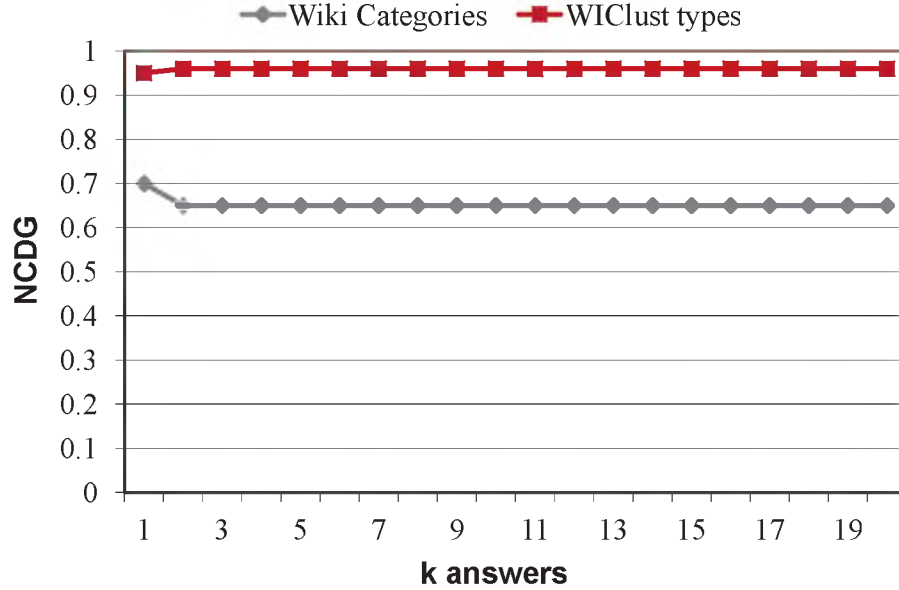


Figure 5.12: WikiQuery results: WIClust types vs. WikiCategories

page structure for all users to follow, Gleim et al. [45] used the page content and layout features to cluster similar Wikipedia pages. Auer and Lehmann [8] used infobox templates and applied manually constructed mappings to assign different template names and attributes into a manually built ontology. Because rules and mappings need to be created manually, this approach requires considerable human intervention. It also misses many template names from the long tail of infrequent template names. YAGO [95], on the other hand, ignored infobox templates. Instead, it made use of the hand-crafted knowledge from the Wikipedia categories to extract concept names, and maps them to the WordNet ontology. However, as we mentioned, this approach has drawbacks. Notably, due to the fact that Wikipedia category names are heterogeneous and folksonomy oriented, and the sometimes inconsistent information from user assignment, it can be difficult to accurately infer entity types. They do obtain 95% accuracy, but for this, they require human intervention to select categories, to define mappings between attributes and concepts, and to correct mistakes that arise from mismatches between Wikipedia and WordNet. In particular, Wikipedia has different types of categories: conceptual categories, administrative category, relational category, and thematic vicinity. In Yago, the administrative and relational categories were excluded by hand. Thematic categories are avoided too. Only the conceptual categories are considered. To distinguish between the conceptual and thematic categories, they employ a shallow linguistic parsing of the category name. Note that this is a limitation of the approach: since the *linguistic heuristic* is language-dependent, heuristics need to be defined for each different language, while the structural similarity is language-independent (i.e., infoboxes are similar for similar types in the same language). Mappings

and synonyms were also specified manually. For example, a user has to write a rule that states that the attributes `born` and `birthday` are mapped to the relation `birthdate`. Last but not least, there were special cases which YAGO had to correct manually. For example, all categories with the head compound `capital` in Wikipedia mean the `capital city`, but the most frequent sense in Word-Net is `financial asset`. YAGO also applied quality controls such as inductive type checking which can introduce more mistakes. For example, the inductive type checking mistook a racing horse for a person because it had a birth date.

Wu et al. [105] used simple heuristics to normalize infobox names and then grouped together infoboxes with the same canonical names. However, using only canonical infobox names is insufficient because they are usually terse, ambiguous, and contain acronyms. For example, it is not trivial to merge template names `Film and Movie`; `Language, Language Family and OS Family`; `Book Series, Novel Series and Game Series`. In order to identify class subsumption, they use YAGO data for training and employ different features, including class-name inclusion, bag-of-word attribute similarity score, and external information from the article edit history. Unlike these approaches, WIClust is unsupervised and relies solely on the structured information from the infoboxes to infer the entity types. As a result, it supports the dynamic nature of Wikipedia and it is also effective at identifying infrequent types.

Like WIClust, Ailon et al. [4] exploited the correlation among data points to cluster them. However, their grouping condition is loose because it simply pulls all the neighbors for each pivot. Thus, it is very sensitive to the grouping threshold. The problem can be attenuated by using the same mechanism we applied in WIClust (Section 5.3.1). Another limitation is the order of selecting pivots, which is key to the quality of the resulting clusters. Since there is no perfect way to identify *all* ambiguous and optional attributes, they can result in low quality clusters. Instead of choosing pivots randomly, WIClust is guided by the magnitude of the negative correlation to initialize the clustering process with a set of more balanced points and iteratively builds the clusters in a more robust way.

Krötzsch et al. [63] suggested that Wikipedia content should be enriched with *semantically interpretable* information, e.g., typed links and page attributes. A barrier to a wide adoption of this suggestions is that it requires additional manual work for editors. WIClust could be used to assist in this task.

To find synonyms, DBpedia [8] and YAGO [95] manually encoded synonyms in their mapping rules, while KOG [105] used canonical attribute names and external knowledge from user edit history. Traditional schema-matching approaches [88] that use label and value similarity to find synonym attributes for similar database schema are not effective in this scenario, because Wikipedia attribute names are terse and data values are not normalized. Similar to [53, 93], we use correlation

as a source of similarity (and dissimilarity) information. However, in order to deal with ambiguous and optional attributes, which make attribute correlation artificially high, we make use of *link types* and combine multiple sources of similarity in a composite prudent schema matcher [81].

5.8 Conclusion

We proposed a new clustering algorithm for organizing Wikipedia infoboxes which is resilient to the skew in the entity distribution and the presence of optional and ambiguous schema attributes. Our algorithm outputs a set of types and corresponding schemata that, as discussed in Section 5.6, can be used to support structured queries over Wikipedia without requiring the user to resolve the inconsistencies and heterogeneity in infoboxes. We have also used `WIClust` to align types of infoboxes defined in different languages [79]. The derived entity types were crucial in the identification interlanguage synonyms, substantially improving the coverage of multilingual queries. Another potential use for `WIClust` would be to enrich DBpedia—it could add new types as well as aid in flagging incorrect type assignments.

As we discussed in Section 5.3.4, `WIClust` is able to refine types, but it derives an essentially flat set of types. We would like to investigate more sophisticated strategies to improve the quality of our clusters and provide a richer taxonomy for the entity by investigating the use of categories and template names in our clustering process.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

Structured data on the Web have been growing at a very fast pace, notably the hidden Web, whose contents typically reside in databases and are only exposed on demand, as users fill out and submit forms, and Wikipedia infoboxes that contain sets of attribute-value pairs that summarize important information about an entity. The availability of documents in multiple languages also opens up new opportunities for querying structured Wikipedia content, and in particular, to enable answers that straddle different languages. Integrating these data can enable advanced queries and make search engines more powerful. In this dissertation, we address the problem of matching a large number of Web-forms and multilingual Wikipedia infoboxes, a necessary step toward integrating these structured data. Another problem that we address is to organize Wikipedia infoboxes in such a way as to discover their entity types and relationships. The commonality among these problems is the data heterogeneity which is compounded by the Web scale and noise from automatic processes or from crowdsourced content. Although different approaches have been proposed, they share an important shortcoming: some require that the input data be clean or preprocessed so as to deal with their heterogeneity. However, the reliance on such preprocessing is problematic for a large heterogeneous dataset, since manual preprocessing is time consuming and expensive, and automated processes in the integration pipeline are error prone. In contrast, our proposed approaches are automatic and *data-driven* processes, which take the benefit from a large number of available data sources, particularly the internal correlation among attributes to naturally reveal the underlying structures from the data such as the attribute correspondences or the entity schematas. The correlation information is also combined with other information in a prudent fashion to minimize error propagation.

In particular, we have presented a set of techniques and tools that overcome these challenges and enable on-the-fly and automatic integration of structured Web data.

- Matching Web-form interfaces: We leverage the availability of a large number of forms to determine correlations among attributes. This correlation is used as a source of similarity. The correlation is combined with our unique features of label similarity and domain value similarity; and to avoid the propagation of matching errors, we prioritize matches with high

confidence. The initial, high-confidence matches are used to resolve uncertain ones and incrementally grow the set of (certain) matches. Our experiments for real datasets show that `FormMatch` obtains high precision and recall without any manual preprocessing; has higher accuracy (from 10% to 68%) than existing holistic approaches; and it is able to identify matches for infrequent attributes that are commonplace in Web-forms [81, 80].

- **Matching multilingual Wikipedia infoboxes:** To support multilingual queries, we proposed `WikiMatch`, a method for identifying mappings between attributes from infoboxes that come from pages in different languages. Our approach leverages *latent semantic* analysis and other kinds of information readily available in Wikipedia to find mappings across multiple infoboxes in a completely automated fashion. Not only can `WikiMatch` be used to find mappings between many language pairs, but it is also effective for languages that are under-represented and lack sufficient training samples. Another important benefit of our approach is that it does not depend on syntactic similarity between attribute names, and thus, it can be applied to language pairs that have distinct morphologies. We have performed an extensive experimental evaluation using a corpus consisting of pages in Portuguese, Vietnamese, and English. We also compared `WikiMatch` against state-of-the-art techniques from data integration [9] and information retrieval [67], as well as to a technique specifically designed to align infobox attributes [20]. The results show that `WikiMatch` outperforms existing approaches in terms of F-measure, and in particular, it obtains substantially higher recall. We also present a case study where we showed that, through the use of the correspondences derived by `WikiMatch`, a multilingual querying system is able to derive higher-quality answers [79].
- **From the problems of matching Web-forms and multilingual Wikipedia infoboxes,** as a step forward, we generalize and propose a general schema-matching framework (`PruSM`) that can find matches for a large number of schematas, given the Web scale and heterogeneity. The `PruSM` framework has been applied to distinct scenarios of matching Web-form interfaces and matching multilingual Wikipedia infoboxes, indicating that `PruSM` is effective for matching large collections of structured Web data.
- **Discovering entity types and relationships for Wikipedia infoboxes:** we leverage the structured information available in Wikipedia infoboxes, and in an attempt to discover the correct schema for a given entity, we group together infoboxes that have similar schemas. We take advantage of the large number of available infobox schemas to compute attribute correlations for each attribute pair, and use them as a source of similarity (and dissimilarity) information to cluster infobox schemata. In particular, we apply a two-pronged approach: before clustering the infoboxes, we first discover the representative attribute sets that are highly correlated

and thus likely candidates for describing an entity and we use these as the basis to group similar infobox schemata together. We also leverage the topological structure of the infoboxes, specifically, the link patterns among the entities to discover meaningful relationships, as well as to reconcile and refine the entity types. Our experiments using over 100,000 infoboxes extracted from Wikipedia show that our approach outperforms other clustering methods, and that it is effective and able to construct an accurate schema for Wikipedia content, even in the presence of noisy, manually edited data. The derived entity clusters have high coverage and quality. A comparison against DBpedia data shows that our clusters are meaningful, cohesive, and include new entity types that are not covered by DBpedia. Furthermore, since this process is automated, it supports the dynamic crowdsourced nature of Wikipedia.

We also note that, since our work deals with *real* data on the Web, there are a few challenges that we have encountered. First and foremost, data on the Web are very heterogeneous, e.g., there is a great variation in how forms are designed or how infoboxes are presented. Because our (large) form collection is automatically gathered by a focused crawler, there is inherent noise from the previous process in the automation pipeline such as domain classification or label extraction which challenges the integration process. Similarly, due to the mass community contribution, the information available in Wikipedia is very heterogeneous, including schema drift, duplicated, generic, or very specific templates. Since we extract the HTML tables, we also deal with the problem of table shifting.¹ The challenges when working with multilingual data include how to process and store the diacritics. In addition, the machine translation and dictionary for multilingual infobox attributes are often insufficient. Last, because the datasets are huge, evaluating the results is difficult. Thus, to evaluate the recall, we have to do the evaluation on a sample set.

6.2 Future Work

As the future work, we suggest the following directions:

- **FormMatch:** Using the match information derived by `FormMatch`, we plan to investigate techniques for automatically filling out the forms and retrieve the contents hidden behind them.
- **WikiMatch:** Our discovered cross-language synonyms is useful for integrating and fusing the multilingual Wikipedia infobox. Our preliminary results in find missing links and entity resolution, especially for languages with different morphology, are very promising. Besides infoboxes, we would like to investigate the effectiveness of `WikiMatch` on other valuable sources of structured data present in Wikipedia, including tables and category links. We

¹some infoboxes are vertically aligned while others are horizontally aligned

would also like to use the discovered cross-language synonyms to build and improve the multilingual faceted search for Wikipedia.

- **PruSM:** Besides matching Web-forms and Wikipedia infoboxes, we would like to integrate more matchers into PruSM and customize it for a wider variety of scenarios and datasets. Because our approach is automated, the results produced can be uncertain or incorrect. To properly deal with this issue during the query evaluation, we plan to explore approaches that take uncertainty into account [34].
- **WIClust:** To improve the quality of our clusters and provide a richer taxonomy for the entity, we would like to investigate the use of categories and template names in our clustering process. Using the WIClust results, we can make suggestions to improve Wikipedia content: There are modification suggestions from the semantic Web community [63, 100] to improve the Wikipedia textual content with semantically interpretable information (typed links and page attributes). Such manual modification is hard and counteracts the ease of use for editors. Our works do not require extra work from users, but such can complement this task. Furthermore, understanding the entity schemata and the derived link patterns, we can combine with Kylin [103] to suggest creating more infoboxes (for pages that follow the link patterns with other infoboxes but do not contain any infobox) and suggest *meta categories* for pages that contain similar entity types. Although our approach to group attributes relies on hard clustering, it treats ambiguous attributes in a special way by creating clusters that share attributes. As our experiments show, high-quality clusters are derived. Nonetheless, it would be interesting to also consider soft clustering strategies and study how they compare to our approach. It would be interesting to use WIClust to discover the entity types and relationships for the *whole* Wikipedia. Besides, using our clustering results as a starting point, we can combine with crowdsourced solutions for improving Wikipedia content. Last, since our clustering approach is schema-centric, it can be applied to cluster Web-form interfaces, Web tables, or other structured entities.

Given the Web data proliferation [69, 104], it is important to design automatic and data-driven processes like our approaches. Machine learning is a powerful tool because it can learn complex hidden patterns from data. Supervised machine learning systems often require users to build a mediated schema and manually construct the semantic mappings in the training examples [32]. Given the heterogeneity of a large number of Web-forms where the importance of different feature varies a lot in different forms and in different domains, creating the training data is expensive. To reduce the burden of training data, we can apply techniques such as active learning to learn from little labeled data [96] or use domain adaptation and transfer learning where we want to deploy the

models learned from some fixed source domain to more different domains [27, 18]. Although human interaction is expensive, it is more accurate. Therefore, it may be better to adopt a collaborative approach between human and machine [99]. Recently, crowdsourcing helps farming out tasks to a large number of users over the web. Although services like Mechanical Turk have opened doors to tap human potential, quality control is also a concern [65].

In this thesis, we consider the problem of discovery implicit structure within the data as well as correspondences across data sources, which are important constituents in the data integration process. In the future, we would like to investigate other constituents in the integration pipeline such as querying, visualization, and real time data delivery.

REFERENCES

- [1] ADAMIC, L., AND ADAR, E. Friends and neighbors on the web. *Social Networks* 25 (2001), 211–230.
- [2] ADAR, E., SKINNER, M., AND WELD, D. S. Information arbitrage across multi-lingual wikipedia. In *Proceedings of Web Search and Data Mining (WSDM)* (2009), pp. 94–103.
- [3] AGRAWAL, R., AND SRIKANT, R. Fast algorithms for mining association rules in large databases. In *Proceedings of the International Conference on Very Large Database (VLDB)* (1994), pp. 487–499.
- [4] AILON, N., CHARIKAR, M., AND NEWMAN, A. Aggregating inconsistent information: Ranking and clustering. *J. ACM* 55 (2008), 23:1–23:27.
- [5] ALEXE, B., CHITICARIU, L., MILLER, R. J., AND TAN, W. C. Muse: Mapping understanding and design by example. In *Proceedings of IEEE International Conference on Data Engineering (ICDE)* (2008), pp. 10–19.
- [6] ARTHUR, D., AND VASSILVITSKII, S. k-means++: The advantages of careful seeding. In *Symposium on Discrete Algorithms (SODA)* (2007), pp. 1027–1035.
- [7] AUER, S., BIZER, C., KOBILAROV, G., LEHMANN, J., AND IVES, Z. Dbpedia: A nucleus for a web of open data. In *Proceedings of the International Symposium on Wearable Computers (ISWC)* (2007), pp. 11–15.
- [8] AUER, S., AND LEHMANN, J. What have innsbruck and leipzig in common? extracting semantics from wiki content. In *Extended Semantic Web Conferences (ESWC)* (2007), pp. 503–517.
- [9] AUMUELLER, D., DO, H. H., MASSMANN, S., AND RAHM, E. Schema and ontology matching with COMA++. In *Proceedings of ACM SIGMOD* (2005), pp. 906–908.
- [10] BAEZA-YATES, R. A., AND RIBEIRO-NETO, B. A. *Modern Information Retrieval*. ACM Press/Addison-Wesley, 1999.
- [11] BALMIN, A., AND CURTMOLA, E. Wikianalytics: Disambiguation of keyword search results on highly heterogeneous structured data. In *WebDB* (2010).
- [12] BARBOSA, L., AND FREIRE, J. Siphoning hidden-web data through keyword-based interfaces. In *Brazilian Symposium on Database (SBBD)* (2004), pp. 309–321.
- [13] BARBOSA, L., AND FREIRE, J. An adaptive crawler for locating hidden-web entry points. In *Proceedings of the International Conference on World Wide Web (WWW)* (2007), pp. 441–450.
- [14] BARBOSA, L., AND FREIRE, J. Combining classifiers to identify online databases. In *Proceedings of the International Conference on World Wide Web (WWW)* (2007), pp. 431–440.

- [15] BARBOSA, L., FREIRE, J., AND DA SILVA, A. S. Organizing hidden-web databases by clustering visible web documents. In *Proceedings of IEEE International Conference on Data Engineering (ICDE)* (2007), pp. 326–335.
- [16] BHATTACHARYA, I., AND GETOOR, L. Collective entity resolution in relational data. *ACM Transaction on Knowledge Discovery from Data* 1, 1 (2007).
- [17] BIZER, C., LEHMANN, J., KOBILAROV, G., AUER, S., BECKER, C., CYGANIAK, R., AND HELLMANN, S. Dbpedia - a crystallization point for the web of data. *JWS* 7, 3 (2009), 154–165.
- [18] BLITZER, J., McDONALD, R., AND PEREIRA, F. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing* (2006), EMNLP '06, pp. 120–128.
- [19] BOLLACKER, K. D., EVANS, C., PARITOSH, P., STURGE, T., AND TAYLOR, J. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of ACM SIGMOD* (2008), pp. 1247–1250.
- [20] BOUMA, G., DUARTE, S., AND ISLAM, Z. Cross-lingual alignment and completion of wikipedia templates. In *Third International Workshop on Cross Lingual Information Access: Addressing the Information Need of Multilingual Societies* (2009), pp. 21–29.
- [21] BOZOVIC, N., AND VASSALOS, V. Two-phase schema matching in real world relational databases. In *International Conference on Data Engineering Workshop (ICDEW)* (2008), pp. 290–296.
- [22] BRANDES, U. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology* 25 (2001), 163–177.
- [23] BRUNK, H. D. *An Introduction to Mathematical Statistics*. 1965.
- [24] BUBECK, S., AND LUXBURG, U. V. Nearest neighbor clustering: A baseline method for consistent clustering with arbitrary objective functions. *J. Mach. Learn. Res.* 10 (June 2009), 657–698.
- [25] CARDOSO, N. GikiCLEF topics and wikipedia articles: Did they blend? In *Multilingual Information Access Evaluation I. Text Retrieval Experiments*, vol. 6241 of *Lecture Notes in Computer Science (LNCS)*. Springer, 2010, pp. 318–321.
- [26] CHANG, K. C.-C., HE, B., AND ZHANG, Z. Toward Large-Scale Integration: Building a MetaQuerier over Databases on the Web. In *Proceedings of Innovative Data Systems Research (CIDR)* (2005), pp. 44–55.
- [27] CORTES, C., MOHRI, M., RILEY, M., AND ROSTAMIZADEH, A. Sample selection bias correction theory. In *Algorithmic Learning Theory (ALT)* (2008), pp. 38–53.
- [28] Dbpedia ontology. <http://www4.wiwiw.fu-berlin.de/dbpedia/dev/ontology.htm>.
- [29] DE MELO, G., AND WEIKUM, G. Towards a universal wordnet by learning from combined evidence. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)* (2009), ACM, pp. 513–522.
- [30] DE MELO, G., AND WEIKUM, G. Menta: inducing multilingual taxonomies from wikipedia. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)* (2010), CIKM '10, ACM, pp. 1099–1108.

- [31] DEERWESTER, S., DUMAIS, S. T., FURNAS, G. W., LANDAUER, T. K., AND HARSHMAN, R. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41, 6 (1990), 391–407.
- [32] DOAN, A., DOMINGOS, P., AND HALEVY, A. Y. Reconciling schemas of disparate data sources: a machine-learning approach. *SIGMOD Rec.* 30, 2 (2001), 509–520.
- [33] DOMSHLAK, C., GAL, A., AND ROITMAN, H. Rank aggregation for automatic schema matching. *IEEE Trans. on Knowl. and Data Eng.* 19, 4 (2007), 538–553.
- [34] DONG, X., HALEVY, A. Y., AND YU, C. Data integration with uncertainty. In *Proceedings of the International Conference on Very Large Database (VLDB)* (2007), pp. 687–698.
- [35] DORNESCU, I. Semantic QA for encyclopaedic questions: *QUAL* in GikiCLEF. In *Proceedings of Cross-Language Evaluation Forum (CLEF)* (2009), pp. 326–333.
- [36] DOS SANTOS, C. T., QUARESMA, P., AND VIEIRA, R. An API for multi-lingual ontology matching. In *Language Resources and Evaluation* (2010).
- [37] DRAGUT, E., FANG, F., SISTLA, P., YU, C., AND MENG, W. Stop word and related problems in web interface integration. In *Proceedings of the International Conference on Very Large Database (VLDB)* (2009), vol. 2, VLDB Endowment, pp. 349–360.
- [38] FAGIN, R., HAAS, L. M., HERNÁNDEZ, M. A., MILLER, R. J., POPA, L., AND VELEGRAKIS, Y. Clio: Schema mapping creation and data exchange. In *Conceptual Modeling: Foundations and Applications* (2009), pp. 198–236.
- [39] FERRANDEZ, S., TORAL, A., FERRANDEZ, I., FERRANDEZ, A., AND MUNOZ, R. Exploiting wikipedia and eurowordnet to solve cross-lingual question answering. *Information Sciences* 179, 20 (2009), 3473–3488.
- [40] FU, B., BRENNAN, R., AND O’SULLIVAN, D. Cross-lingual ontology mapping - an investigation of the impact of machine translation. In *Asian Semantic Web Conference (ASWC)* (2009), pp. 1–15.
- [41] FUXMAN, A., AND MILLER, R. J. Schema mapping. In *Encyclopedia of Database Systems*. 2009, pp. 2481–2488.
- [42] GAL, A. Managing uncertainty in schema matching with top-k schema mappings. In *J. Data Semantics VI* (2006), pp. 90–114.
- [43] GAL, A. *Uncertain Schema Matching*. Morgan & Claypool Publishers, 2011.
- [44] GikiCLEF Cross-language Geographic Information Retrieval from Wikipedia. <http://www.linguatca.pt/GikiCLEF>.
- [45] GLEIM, R., MEHLER, E., AND DEHMER, M. Web corpus mining by instance of wikipedia. In *European Chapter of the Association for Computational Linguistics (EACL)* (2007).
- [46] Google translator. <http://www.google.com/translate>.
- [47] Google Base. <http://base.google.com/>.
- [48] GRAVANO, L., GARCÍA-MOLINA, H., AND TOMASIC, A. Gloss: text-source discovery over the internet. *ACM Trans. Database Syst.* 24, 2 (1999), 229–264.

- [49] HAI DO, H., AND RAHM, E. Coma - a system for flexible combination of schema matching approaches. In *Proceedings of the International Conference on Very Large Database (VLDB)* (2002), pp. 610–621.
- [50] HASSANZADEH, O., CHIANG, F., LEE, H. C., AND MILLER, R. J. Framework for evaluating clustering algorithms in duplicate detection. *PVLDB* 2 (2009), 1282–1293.
- [51] HAVELIWALA, T. H. Scalable techniques for clustering the web. In *In Proc. of the WebDB Workshop* (2000), pp. 129–134.
- [52] HE, B., AND CHANG, K. C.-C. Statistical schema matching across web query interfaces. In *Proceedings of ACM SIGMOD* (2003), ACM, pp. 217–228.
- [53] HE, B., AND CHANG, K. C.-C. Automatic complex schema matching across web query interfaces: A correlation mining approach. *Transactions on Database Systems (TODS)* 31, 1 (2006), 346–395.
- [54] HE, B., TAO, T., AND CHANG, K. C.-C. Organizing structured web sources by query schemas: a clustering approach. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)* (2004), pp. 22–31.
- [55] HE, H., AND MENG, W. Wise-integrator: An automatic integrator of web search interfaces for e-commerce. In *Proceedings of the International Conference on Very Large Database (VLDB)* (2003), pp. 357–368.
- [56] HU, W., QU, Y., AND CHENG, G. Matching large ontologies: A divide-and-conquer approach. *Data and Knowledge Engineering* 67, 1 (2008), 140–160.
- [57] JÄRVELIN, K., AND KEKÄLÄINEN, J. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)* (2000), pp. 41–48.
- [58] JÄRVELIN, K., AND KEKÄLÄINEN, J. Cumulated gain-based evaluation of ir techniques. *Transactions on Information Systems (TOIS)* 20 (2002), 422–446.
- [59] KANG, J., AND NAUGHTON, J. F. On schema matching with opaque column names and data values. In *Proceedings of ACM SIGMOD* (2003), ACM Press, pp. 205–216.
- [60] KASNECI, G., RAMANATH, M., SOZIO, M., SUCHANEK, F. M., AND WEIKUM, G. Star: Steiner-tree approximation in relationship graphs. In *Proceedings of IEEE International Conference on Data Engineering (ICDE)* (2009), pp. 868–879.
- [61] KASNECI, G., RAMANATH, M., SUCHANEK, F., AND WEIKUM, G. The yago-naga approach to knowledge discovery. *SIGMOD Rec.* 37, 4 (2008), 41–47.
- [62] KASNECI, G., SUCHANEK, F., IFRIM, G., RAMANATH, M., AND WEIKUM, G. NAGA: Searching and ranking knowledge. In *Proceedings of IEEE International Conference on Data Engineering (ICDE)* (2008), pp. 953–962.
- [63] KRTZSCH, M., VRANDECIC, D., VR, D., AND VLKEL, M. Wikipedia and the semantic web - the missing links. In *Wikimania* (2005).
- [64] KUMARAN, A., SARAVANAN, K., DATHA, N., ASHOK, B., AND DENDI, V. Wikibabel: A wiki-style platform for creation of parallel data. In *Asian Federation of Natural Language Processing (ACL/AFNLP)* (2009), pp. 29–32.

- [65] LEASE, M. On Quality Control and Machine Learning in Crowdsourcing. In *Proceedings of the 3rd Human Computation Workshop (HCOMP) at the Conference on Artificial Intelligence (AAAI)* (2011), pp. 97–102. Separately refereed and accepted for encore presentation at the AAAI Spring Symposium 2012: Wisdom of the Crowd.
- [66] LIBEN-NOWELL, D., AND KLEINBERG, J. The link prediction problem for social networks. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*.
- [67] LITTMAN, M., DUMAIS, S. T., AND LANDAUER, T. K. Automatic cross-language information retrieval using latent semantic indexing. In *Proceedings of Cross-Language Information Retrieval (CLIR)* (1998), pp. 51–62.
- [68] LIU, B. *Web Data Mining. Exploring Hyperlinks, Contents, and Usage Data*. Springer, 2007.
- [69] MADHAVAN, J., COHEN, S., DONG, X. L., HALEVY, A. Y., JEFFERY, S. R., KO, D., AND YU, C. Web-scale data integration: You can afford to pay as you go. In *Proceedings of Innovative Data Systems Research (CIDR)* (2007), pp. 342–350.
- [70] MADHAVAN, J., KO, D., KOT, L., GANAPATHY, V., RASMUSSEN, A., AND HALEVY, A. Y. Google’s deep web crawl. *PVLDB* 1, 2 (2008), 1241–1252.
- [71] MAGNINI, B., GIAMPICCOLO, D., FORNER, P., AYACHE, C., JIKOUN, V., OSENOVA, P., PEÑAS, A., ROCHA, P., SACALEANU, B., AND SUTCLIFFE, R. F. E. Overview of the CLEF 2006 multilingual question answering track. In *CLEF* (2006), pp. 223–256.
- [72] MANNING, C. D., RAGHAVAN, P., AND SCHÜTZE, H. *Introduction to Information Retrieval*. Cambridge Univ. Press, 2008.
- [73] MELNIK, S., GARCIA-MOLINA, H., AND RAHM, E. Similarity flooding: A versatile graph matching algorithm and its application. In *Proceedings of IEEE International Conference on Data Engineering (ICDE)* (2002), pp. 117–128.
- [74] MITCHELL, T. M. *Machine Learning*, 1 ed. McGraw-Hill, Inc., New York, NY, USA, 1997.
- [75] NGUYEN, D., OVERWIJK, A., HAUFF, C., TRIESCHNIGG, D., HIEMSTRA, D., AND DE JONG, F. Wikitranslate: Query translation for cross-lingual information retrieval using only wikipedia. In *Evaluating Systems for Multilingual and Multimodal Information Access* (2009), vol. 5706 of *Lecture Notes in Computer Science (LNCS)*, pp. 58–65.
- [76] NGUYEN, H., FUXMAN, A., PAPARIZOS, S., FREIRE, J., AND AGRAWAL, R. Synthesizing products for online catalogs. *Proceedings of the International Conference on Very Large Database (VLDB)* 4, 7 (2011), 409–418.
- [77] NGUYEN, H., NGUYEN, T., AND FREIRE, J. Learning to extract form labels. In *Proceedings of the International Conference on Very Large Database (VLDB)* (2008), vol. 1, VLDB Endowment, pp. 684–694.
- [78] NGUYEN, H., NGUYEN, T., NGUYEN, H., AND FREIRE, J. Querying Wikipedia Documents and Relationships. In *WebDB* (2010).
- [79] NGUYEN, T., MOREIRA, V., NGUYEN, H., NGUYEN, H., AND FREIRE, J. Multilingual schema matching for wikipedia infoboxes. In *Proceedings of the International Conference on Very Large Database (VLDB)* (2012), vol. 5, VLDB Endowment, pp. 133–144.

- [80] NGUYEN, T., NGUYEN, H., AND FREIRE, J. Prudent schema matching for a large number of web-form interfaces. Tech. rep., University of Utah, Salt Lake, UT, 2009.
- [81] NGUYEN, T., NGUYEN, H., AND FREIRE, J. PruSM: A prudent schema matching strategy for web-form interfaces. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)* (2010), ACM, pp. 1385–1388.
- [82] NGUYEN, T., NGUYEN, H., MOREIRA, V., AND FREIRE, J. Clustering wikipedia infoboxes to discover their types. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)* (2012), ACM, p. To appear.
- [83] NING TAN, P. Selecting the right interestingness measure for association patterns. In *Proceedings of Knowledge discovery and data mining (KDD)* (2002), pp. 32–41.
- [84] Ontology alignment evaluation initiative. <http://oei.ontologymatching.org>.
- [85] OH, J.-H., KAWAHARA, D., UCHIMOTO, K., KAZAMA, J., AND TORISAWA, K. Enriching multilingual language resources by discovering missing cross-language links in wikipedia. In *Proceedings of Web Intelligence and Intelligent Agent Technology (WI-IAT)* (2008), vol. 1, pp. 322–328.
- [86] PEI, J., HONG, J., AND BELL, D. A robust approach to schema matching over web query interfaces. In *Proceedings of IEEE International Conference on Data Engineering (ICDE)* (2006), pp. 46–55.
- [87] POTTHAST, M., STEIN, B., AND ANDERKA, M. A Wikipedia-based multilingual retrieval model. In *European Conference on Information Retrieval (ECIR)* (2008), pp. 522–530.
- [88] RAHM, E., AND BERNSTEIN, P. A. A survey of approaches to automatic schema matching. *The VLDB Journal* 10, 4 (2001), 334–350.
- [89] SCHÖNHOFEN, P., BENCZÚR, A., BÍRÓ, I., AND CSALOGÁNY, K. Cross-language retrieval with wikipedia. In *Advances in Multilingual and Multimodal Information Retrieval* (2008), pp. 72–79.
- [90] SHI, J., AND MALIK, J. Normalized cuts and image segmentation. *Proceedings of Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 22, 8 (2000), 888–905.
- [91] SHVAIKO, P., AND SHVAIKO, P. A survey of schema-based matching approaches. *Journal on Data Semantics* 4 (2005), 146–171.
- [92] SORG, P., AND CIMIANO, P. Enriching the crosslingual link structure of wikipedia - a classification-based approach. In *WikiAI* (2008), pp. 1–6.
- [93] SU, W., WANG, J., AND LOCHOVSKY, F. Holistic query interface matching using parallel schema matching. In *Proceedings of Extending Database Technology (EDBT)* (2006), pp. 77–94.
- [94] SUCHANEK, F. M., ABITEBOUL, S., AND SENELLART, P. Paris: probabilistic alignment of relations, instances, and schema. In *Proceedings of the International Conference on Very Large Database (VLDB)* (2011), vol. 5, pp. 157–168.
- [95] SUCHANEK, F. M., KASNECI, G., AND WEIKUM, G. Yago: A Core of Semantic Knowledge. In *Proceedings of the International Conference on World Wide Web (WWW)* (2007), pp. 697–706.

- [96] TONG, S. *Active Learning: Theory and Application*. PhD thesis, Stanford University, 2001.
- [97] UDUPA, R., AND KHAPRA, M. Improving the multilingual user experience of wikipedia using cross-language name search. In *Proceedings of Association for Computational Linguistics: Human Language Technologies (ACL-HLT)* (2010), pp. 492–500.
- [98] VAN DONGEN, S. M. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, The Netherlands, 2000.
- [99] VAN DONGEN, S. M. *Active Learning and Crowdsourcing for Machine Translation in Low Resource Scenarios*. PhD thesis, Carnegie Mellon University, 2011.
- [100] VÖLKEL, M., KRÖTZSCH, M., VRANDECIC, D., HALLER, H., AND STUDER, R. Semantic wikipedia. In *Proceedings of the International Conference on World Wide Web (WWW)* (2006), pp. 585–594.
- [101] WANG, H., TAN, S., TANG, S., YANG, D., AND TONG, Y. Identifying indirect attribute correspondences in multilingual schemas. In *Proceedings of Conference on Database and Expert Systems (DEXA)* (2006), pp. 652–656.
- [102] WANG, J., WEN, J., LOCHOVSKY, F., AND MA, W. Instance-based schema matching for web databases by domain-specific query probing. In *Proceedings of the International Conference on Very Large Database (VLDB)* (2004), pp. 408–419.
- [103] WELD, D. S., HOFFMANN, R., AND WU, F. Using wikipedia to bootstrap open information extraction. *SIGMOD Rec.* 37 (2009), 62–68.
- [104] Wikimedia traffic analysis report. <http://stats.wikimedia.org/wikimedia/squids/SquidReportPageViewsPerCountryOverview.htm>.
- [105] WU, F., AND WELD, D. S. Automatically refining the wikipedia infobox ontology. In *Proceedings of the International Conference on World Wide Web (WWW)* (2008), pp. 635–644.
- [106] WU, W., AND DOAN, A. Webiq: Learning from the web to match deep-web query interfaces. In *Proceedings of IEEE International Conference on Data Engineering (ICDE)* (2006), pp. 44–54.
- [107] WU, W., YU, C., DOAN, A., AND MENG, W. An interactive clustering-based approach to integrating source query interfaces on the deep web. In *Proceedings of ACM SIGMOD* (2004), pp. 95–106.
- [108] ZHANG, Z., HE, B., AND CHANG, K. Understanding web query interfaces: best-effort parsing with hidden syntax. In *Proceedings of ACM SIGMOD* (2004), pp. 107–118.
- [109] ZHAO, P., HAN, J., AND SUN, Y. P-rank: a comprehensive structural similarity measure over information networks. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)* (2009), pp. 553–562.